AD-A207 802

Information Requirements and the Implications for Parallel Computation

by

Patrick Haven Worley

Department of Computer Science

Stanford University
Stanford, California 94305

DTIC
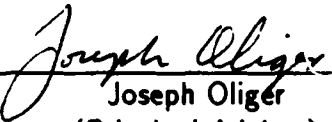ELECT
MAY 0 8 1989
S D
H

89 2 16 209

INFORMATION REQUIREMENTS AND THE
IMPLICATIONS FOR PARALLEL COMPUTATION

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
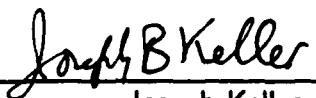
By
Patrick Haven Worley
June 1988

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.
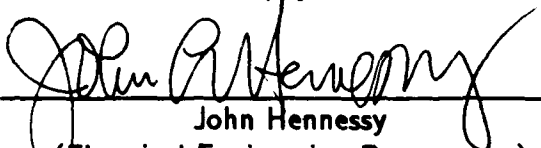
_____
Joseph Oliger
(Principal Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

_____
Joseph Keller
(Mathematics Department)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

_____
John Hennessy
(Electrical Engineering Department)

Approved for the University Committee on Graduate Studies:

_____
Dean of Graduate Studies

iii

Abstract

The best serial algorithms for numerically approximating the solution of a linear partial differential equation (PDE) exploit knowledge of the solution operator. This dissertation describes how the solution operator also influences the behavior of parallel algorithms.

Approximating the solution at a single location in the problem domain is considered. We derive a lower bound on the error in the approximation that is a function of the amount of data used and the smoothness of the data functions. From this we derive a lower bound on the parallel complexity of algorithms that approximate the solution. The lower bound is a linear function of $\log \epsilon^{-1}$, where ϵ is an upper bound on the error. Thus the parallel complexity increases as ϵ decreases, independent of the number of processors used. We also construct an algorithm whose parallel complexity is of this form, proving that the form of the bound is the best possible.

The execution time of parallel algorithms is a function of both the communication costs and the parallel complexity. We describe bounds on the communication costs of parallel algorithms that are functions of the distance between collaborating processors. If the interconnection network of a multiprocessor is a d dimensional grid, then we prove that the execution time of algorithms that approximate the solution is bounded from below by a linear function of $\epsilon^{-\frac{\gamma}{d+1}}$, where γ is a positive constant determined by the smoothness of the data functions. Thus, for small ϵ, the communication costs are the dominant constraints on the optimal performance.

Acknowledgements

QUALITY
INSPECTED
2

v

Accession For	
NTIS GRA&I	☑
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

Contents

List of Figures

Chapter 1

Introduction

The best serial algorithms for numerically approximating the solution of a partial differential equation (PDE) exploit knowledge of the behavior of the solution of the PDE. The design of parallel algorithms is a more complicated process than the design of serial algorithms, and the performance of a parallel algorithm is strongly influenced by the architecture of the multiprocessor used. This work describes ways in which the data dependence of the solution of a well-posed linear scalar PDE determines bounds on the effectiveness of parallel algorithms and architectures.

We assume that a Green's function of a particular form exists for the PDE. We assume that the numerical methods use values of the data functions sampled at locations in their respective domains, and calculate values of the solution at locations in its domain. We assume that the data functions satisfy conditions suitable for bounding, a priori, the error of an mth order accurate approximation to the PDE, for some fixed positive integer m. We assume that the error introduced by a numerical algorithm is required to be less than some given tolerance ϵ for each solution value. These assumptions are described in more detail later in this chapter. Given these assumptions, we describe the following result:

- We calculate a lower bound on the parallel complexity of any parallel algorithm that satisfies the above assumptions. This lower bound is a linear function of $\log_2 \epsilon^{-1}$. Therefore, the execution time required to approximate the solution of the PDE must grow if the error bound decreases, no matter how many processors are available. We also construct an algorithm whose parallel complexity has the same form, proving

1

that the form of the lower bound is the best possible. Finally, we show that the time spent moving data between processors can constrain the execution time even more strongly than the lower bound on the parallel complexity. For example, if the interconnection network of a multiprocessor is a d dimensional mesh, then the execution time is bounded from below by a negative power of ϵ, independent of the number of processors that are available.

The dissertation is organized in the following manner. The rest of this chapter describes our models for multiprocessor architectures, parallel algorithms, and numerical algorithms for approximating solutions to linear PDEs. Chapter 2 describes the data dependency analysis, and properties of the results that will be used in later chapters. Chapter 3 describes lower bounds on the execution time of parallel algorithms as the problem and the multiprocessor grow in size. Chapter 4 describes the bounds derived in the two previous chapters for example problems, and discusses generalizing the assumptions so as to include a larger class of problems. Chapter 5 summarizes the work presented here, and briefly discusses some generalizations of the results. Appendices contain details left out of the main presentation.

1.1   Multiprocessor Architectures

We are primarily interested in MIMD[1] multiprocessors, and in modelling parallelism at the level of concurrent execution of floating point operations. This bias is reflected in the following multiprocessor model. Most of our results apply to a larger class of multiprocessors and parallel algorithms, for an appropriate interpretation.

1.1.1   Multiprocessor Model

Each multiprocessor is made up of memory, processors, and unidirectional communication channels. The multiprocessor is represented by a labelled directed graph[2], (V, E). Each

---

[1] *Multiple Instruction Multiple Data* is one category of Flynn's multiprocessor taxonomy [Fly72]. If a multiprocessor is in MIMD, then the instruction a processor is executing and the data it is using can be different from those of other processors at any given moment.

[2] See p.50 of [AHU74]

vertex, $v_i \in V$, of the graph is either a memory or a processor. Each edge, $e_j \in E$, is a unidirectional communication channel between two vertices, permitting the *source* vertex to send data to the *destination* vertex. The subscript of the source vertex of edge e_j is $s(j)$. The subscript of the destination vertex of edge e_j is $d(j)$. A vertex v_1 is *connected* to a vertex v_2 if there exists an edge for which v_1 is the source and v_2 is the destination. The multiprocessor uses a fixed length floating point number representation for real numbers. This representation is assumed to be sufficient to approximate the solution of a given PDE to within some given error tolerance. The components of the multiprocessor are characterized by their abilities to manipulate these floating point numbers.

Each memory vertex v_i is labelled by two nonnegative numbers. The capacity c_i is the number of floating point numbers the memory can hold. The access time a_i is the time it takes to recall a floating point number from the memory.

Each processor is a *serial processor*, which we define to be a processor that calculates floating point operations sequentially. Each processor vertex v_i is labelled by a positive real number indicating the time it takes to add two floating point numbers together, $f_{i,(+)}$. This is used as the standard unit of computation. All floating point operations are computed by the composition of operators from some given set of binary and unary floating point operators. We assume that negation is the only unary floating point operation among this set of primitives. We also assume that the execution time of any binary floating point operation is greater than or equal to the execution time of a binary floating point addition.

Each edge e_j is labelled by two positive numbers. The bandwidth b_j is the number of floating point numbers that can be transmitted over the channel during some given unit of time τ. The transmission time t_j is the time it takes to send a single floating point number from $v_{s(j)}$ to $v_{d(j)}$ using communication channel e_j. These values may include the overhead of starting a transmission between the vertices, and so b_j may not be τ/t_j. The values will depend on the communication protocol.

Many of the assumptions in this model are merely to establish a concrete model, and are easily generalized. For example, the restriction to binary primitive floating point operators is a reasonable assumption, but we only want to establish an upper bound on the number of operands of the primitive floating point operators.

1.1.2   Communication Capabilities

The performance of a parallel algorithm is strongly affected by the ability of the multi-processor to move data between vertices. Define a *path* P to be a sequence of edges, $\{e_{j_l} \mid l = 1, \ldots, p\}$, such that the destination vertex of edge e_{j_l} is the source vertex for edge $e_{j_{l+1}}$. Define the *length* of the path P, $L(\mathrm{P})$, to be the sum of the transmission times along this path,

$$L(\mathrm{P}) = \sum_{l=1}^{p} t_{j_l} \ .$$

$L(\mathrm{P})$ is then the time required to send a single floating point number along the path.

Define the *distance* from vertex v_1 to vertex v_2, $D(v_1, v_2)$, to be the length of the path of minimum length starting at v_1 and ending at v_2. Define $D(v, v)$ to be zero. $D(v_1, v_2)$ is the minimum amount of time it takes to send a single floating point number from v_1 to v_2.

The *diameter* of a subset of the vertices of the graph, $V' \subset V$, is the maximum distance between two vertices of the subset,

$$diam(V') = \max_{v, w \in V'} D(v, w) \ .$$

A *center* of this subset is a vertex that minimizes the maximum distance between itself and other vertices in the subset,

$$C = \{c \mid c \in V', \ \max_{w \in V'} D(c, w) = \min_{v \in V'} \max_{w \in V'} D(v, w)\} \ .$$

The *radius* of the subset is this distance,

$$rad(V') = \min_{v \in V'} \max_{w \in V'} D(v, w) \ .$$

The radius satisfies

$$diam(V')/2 \le rad(V') \le diam(V') \ .$$

If the vertices in a set V' are all collaborating in a calculation, then the radius is a lower bound on the time spent moving data between vertices during the calculation.

> *Example:* Assume that a calculation produces one floating point number, that one vertex of a subset V' has been designated to hold this result, and that

each vertex in V' is in sole possession of information crucial to the calculation. Then a lower bound on the time spent in moving this information through the communication channels is $rad(V')$. The vertex that has been designated to hold the result cannot be closer than $rad(V')$ to all of the other vertices of the set, and some datum used by the result will take at least this long to arrive at the designated vertex.

1.1.3 Example Architectures

Most multiprocessor architectures currently being investigated have fairly simple graphs, with essentially homogeneous processor and memory capabilities. See Feng [Fen81] for a survey of some of the graphs that have been considered. The following examples are common designs, each of whose behavior is representative of a class of architectures. All of the examples can be described as undirected graphs. If an edge exists from v_i to v_j, then an edge with the same parameters also exists from v_j to v_i. We will refer to this pair as a single edge in the following discussion. Additionally, all processor, memory, and communication channel capabilities are the same unless otherwise noted. The values for the capacity, the access time, the floating point addition time, the bandwidth, and the transmission time are denoted by c, a, $f_{(+)}$, b, and t respectively.

- *Fully Connected.* The most powerful communications network has a graph that is a clique. That is, every vertex is connected to every other vertex. The diameter of any subset of the multiprocessor is t.

- *Star.* One multiprocessor architecture suitable for small numbers of processors can be modelled as N processors connected to a single memory in a star topology. That is, there are N communication channels, and each channel connects a different processor with the memory. The diameter of any subset of the multiprocessor containing at least two processors is $2t$.

- *Distributed Shared Memory.* Both of the previous examples have the property that all memories are equidistant from all processors. We will call an architecture with

this property a *shared memory* architecture.[3] While the two previous examples satisfy this condition, they are expensive to implement for large numbers of processors. A less expensive shared memory architecture replaces the single large memory in the previous examples with many smaller memories and a distinct interconnection network. The vertices of the network are themselves small capacity memories, *switches*, that accept and retransmit data. Each processor and memory external to this interconnection network has one edge into the network. The interconnection network provides paths from any processor to any external memory. Assume that there are N processors and N external memories, where $N = 2^k$ for some positive integer k. If the interconnection network is implemented as an Omega network [HB84] [GGK*83], then there are $.5 \cdot N \log_2 N$ switches, each of which has 4 impinging edges. The distance between any processor and external memory is $t \cdot \log_2 N$. If all communication between processors is via external memory, then the diameter of a subset of vertices containing at least two processors is $2t \cdot \log_2 N$. If interprocessor communication is direct, then a subset of K processors can have a diameter as small as $t \cdot \log_2 K$.

• *d Dimensional Array.* We call an architecture a *local memory* architecture if each memory is connected to only one vertex, a processor, and each processor is connected to only one memory. Thus, each processor has quick access to its local memory, but at the cost of slower access to the others. A common example is a d dimensional array of processor-memory pairs. Each processor is connected to up to $2d$ other processors, forming a d dimensional array. Assume that the array has equal length sides and N processor-memory pairs. Then the diameter of a subset of K processors is no more than $dt \cdot (N^{1/d} - 1)$, and no less than $dt \cdot (K^{1/d} - 1)$. The maximum is the diameter of the multiprocessor. The minimum is achieved by a subset of vertices and edges that is a d dimensional array with $K^{1/d}$ processors on a side.

---

[3]Currently, the most common shared memory multiprocessor architectures can be described as N processors and a single memory connected by a shared bus. The graph of this particular architecture can't be represented using the above model, but the communication capabilities of this network can be considered to fall somewhere between that of the fully connected network and that of the star network.

- *Hypercube.* If the dimension of an array of N processors with equal length sides is $\log_2 N$, then the graph is of a $\log_2 N$ dimensional binary hypercube[4]. Each processor has $\log_2 N$ edges. The diameter of a subset of K processors is between $t \cdot \log_2 N$ and $t \cdot \log_2 K$. The lower bound is associated with a subset of vertices and edges that approximates a $\log_2 K$ dimensional hypercube, or whose complement approximates a $\log_2 (N - K)$ dimensional hypercube.

The difficulty and cost of constructing multiprocessors based on these designs can vary dramatically. As the number of processors grows, more and more of the examples become infeasible. Ultimately, the three dimensionality of the physical world will limit feasible architectures to something that can be embedded in some variant of a three dimensional grid of processors. Both technological limits and basic physical laws constrain how small the processors and memories can be [MC80], and the volume taken up by a multiprocessor must increase as the number of devices increase.[5] Thus, the speed of light will constrain the speed of communications between the devices in the multiprocessor [Kro85]. But these limitations are not the only difficulties associated with these architectures. For the distributed shared memory architecture example, increasing the number of processors increases the minimum number of edges data must travel over when being sent between any processor and external memory. This necessarily increases the communication time when t is kept fixed. For fully connected, star, and hypercube based architectures, the number of input/output ports on a single device grows as the number of processors increase. This increases device size, cost, and complexity.

1.2 Parallel Algorithms

We model an algorithm as a partially ordered set of *instructions* of the form

$$y = flop(x_1, \ldots, x_n) \ .$$

---

[4]See Seitz [Sei85] for a description of a particular hypercube based multiprocessor architecture.
[5]Similar arguments constrain how small $f_{(+)}$ can be.

flop is a floating point operation, y is a floating point variable, and $\{x_1, \ldots, x_n\}$ are floating point constants and variables. If a floating point variable is used by two different instructions, and if one of the instructions changes the value of that variable, then the partial order specifies a precedence relationship between them. These are the only relationships established by the partial order.[6]

This model of an algorithm ignores many of the details usually found in algorithms. In particular, integer arithmetic and instructions controlling conditional execution are not represented. But the time spent executing floating point operations generally dominates the total execution time of algorithms for numerically approximating the solution of PDEs. Moreover, this model is sufficient for establishing the lower bounds described in section 1.2.2.

We define the *serial computational complexity* of an algorithm, C_s, to be the time spent calculating the floating point operations on some standard serial processor. We will often simply refer to this as the *serial complexity*. See [Kro85] and [AHU74] for other definitions of serial complexity. The standard processor is assumed to satisfy the assumptions made in the previous section about the processors in the multiprocessor. All sequential orderings of the instructions of an algorithm that are consistent with the partial ordering will have the same serial complexity, and will produce the same results when executed on a serial processor. Therefore, we will also refer to the partially ordered set of instructions as a *serial algorithm*. By the assumptions made on the processor, the serial complexity is a weighted sum of the number of floating point operations. The weights depend on the specifics of the standard processor.

A parallel implementation of an algorithm on a multiprocessor specifies when and on which processor each instruction is executed, where the data is initially assigned, where each intermediate and final result is stored, and what communication takes place during the execution of the algorithm. We will refer to this information as the *scheduling* of the algorithm. A scheduling is *well-defined* if it is compatible with the partial order's precedence relationships, and if all demands made on the processors, memories, and communication channels are within their capabilities. We define a *parallel algorithm* to be

---

[6]Thus, the algorithm can be represented by a data flow graph. See [HB84, pages 740–744].

the union of a serial algorithm, a multiprocessor architecture, and a well-defined scheduling. Thus, we associate a deterministic serial algorithm with each parallel algorithm. In practice, the serial algorithm may be a function of the data. This can make determining the serial algorithm difficult, especially for chaotic parallel algorithms [CM69] [Bau78]. But we can still analyze characteristics of the parallel algorithm by establishing necessary properties of the associated serial algorithm.

1.2.1 Parallel Algorithm Costs

We define the *cost*, T_p, to be the time it takes to execute a parallel algorithm on a multiprocessor. Unlike serial algorithms, the cost of a parallel algorithm is not well approximated by a weighted sum of the number of floating point operations. Instead, there are two distinct costs associated with an efficient parallel algorithm, *parallel computational complexity* and *communication cost*.

Parallel computational complexity

The parallel computational complexity is the amount of time during which at least one of the processors is busy calculating the instructions of the serial algorithm. We will also refer to the parallel computational complexity as simply the *parallel complexity*, or the *parallel computation cost*. See [Kro85] for other definitions of parallel complexity. If the multiprocessor has N identical processors, then the parallel complexity, C_p, is bounded from below by[7]

$$C_p \geq \left\lceil \frac{C_s}{N} \right\rceil \ . \tag{1.1}$$

The serial complexity in this expression is based on using one of the N processors as the standard serial processor. For this bound to be achieved, all of the processors must be busy all of the time. Even if communication is free, i.e. $a = 0$, $t = 0$, and $b = \infty$, few algorithms have N independent operations to be calculated throughout their execution. The partial order can be examined to determine the maximum number of independent operations that can be calculated at any one point in the algorithm. The schedule may further limit the parallelism.

---

[7] $\lceil x \rceil$ represents the smallest integer greater than or equal to x.

Measures of the parallelism in the parallel algorithm are the *computational speed-up,*

$$S_p = \frac{C_s}{C_p}$$

and the *computational efficiency,*

$$E_p = \frac{S_p}{N} \quad .$$

The computational speed-up represents how much faster the parallel algorithm can be executed than the serial algorithm when communication is free. It is always less than or equal to N. If all the processors are identical, then the computational efficiency measures the average amount of time each of the processors is busy computing. It is always less than or equal to 1.

Example: Consider a multiprocessor with $N/2$ processors, each of which calculates a floating point addition in time $f_{(+)}$. Then the summation of N floating point numbers,

$$\sum_{i=1}^{N} \alpha_i \quad ,$$

requires at least time $f_{(+)} \cdot \lceil \log_2 N \rceil$. Add is a binary operation, and each time step of length $f_{(+)}$ replaces M existing summands by at least $M/2$ results. These results are the summands for the next step.[8]

The best serial algorithm has a serial complexity of $f_{(+)} \cdot (N - 1)$ on one of these processors. Thus, the computational speed-up is bounded from above by $N/\lceil \log_2 N \rceil$ for a parallel algorithm based on this serial algorithm. The computational efficiency is bounded from above by $2/\log_2 N$ when $N/2$ processors are used. The poor computational efficiency reflects the fact that, after each step, half of the processors active previously have no more work to do. If fewer processors are used, then the computational efficiency increases. But then the computational speed-up decreases and the parallel complexity increases.

---

[8]See Lemma 1 in Kuck [Kuc78, page 95].

Communication cost

The communication cost, W_p, is the amount of time during which data is being moved between the processors and memories of the multiprocessor. This includes the time spent accessing a memory location and packaging the data for transmission. The achievable parallel complexity is constrained by the number of processors in the multiprocessor and by the serial algorithm. The communication cost is additionally constrained by the graph of the multiprocessor. For example, the maximum computational speed-up in the summation problem is achieved by using $N/2$ processors for an N term sum. Since each processor produces an intermediate result that is needed to compute the final sum, the radius of the $N/2$ processor subset of the multiprocessor will bound the communication cost from below.

Effective scheduling

Most other reasonable costs, like integer arithmetic or overhead for synchronization, are easily included in the model as communication or parallel computation costs. An unreasonable cost is incurred when the schedule unnecessarily augments the partial order of the serial algorithm, adding extra delays. For a given assignment of instructions to processors and data to memories, we define a well-defined schedule to be *effective* if:

- a processor executes an instruction whenever its operands are available, the processor is not currently executing another instruction or communication request, and the partial order of the serial algorithm is not violated by doing so.

- when a processor is blocked from executing any more instructions due to a lack of data, and that data exists, the data is moved to the processor as quickly as possible.

We henceforth restrict ourselves to parallel algorithms with effective schedules. An effective schedule is not necessarily a good one, but having one ensures that some part of the multiprocessor is actively computing or communicating at all times.

Total cost

A lower bound on T_p is [9]

$$\max\{C_p, W_p\} \quad .$$

But a processor cannot calculate a floating point operation until the operands are available. If the operands must be drawn from some other device, then the calculation of the operation is blocked until they arrive. If the schedule is effective, then an upper bound on the cost of the parallel algorithm is

$$T_p \leq C_p + W_p \quad .$$

We define the *speed-up* and the *efficiency* to be the corresponding measures for the total cost of a parallel algorithm [Kuc78] , i.e.

$$S_t = \frac{C_s}{T_p}$$

and

$$E_t = \frac{S_t}{N}$$

respectively. N is again the number of processors.

Computation bound algorithms

A parallel algorithm is *computation bound* if the communication cost is no more than the parallel complexity, $W_p \leq C_p$. If a parallel algorithm is computation bound for a multiprocessor architecture, then the architecture is adequate in the sense that the partial order of the corresponding serial algorithm determines the dominant part of the cost. This condition is also a useful tool for deciding how many processors to use to solve a problem. For a given multiprocessor architecture, it is common for the parallel complexity to be decreasing and the communication cost to be increasing as the number of processors

---

[9]We will use $\max\{a_1, \ldots, a_n\}$ as an alternative notation for

$$\min_{a \in \{a_1, \ldots, a_n\}} a \quad .$$

Similar notation will be used for the minimum element in a set.

used to execute an algorithm increases. This describes the behavior of a set of parallel algorithms, one for each number of processors, and will only hold over a range of numbers of processors. The largest number of processors for which the corresponding parallel algorithm is computation bound represents an estimate of the number of processors to use to minimize the cost. If $W_p = C_p$, then the cost of the corresponding parallel algorithm is at most twice that of the optimal, and will usually be much better than that. A similar estimate can be used to evaluate a type of architecture as both the problem size and the number of processors grow.

> *Example:* The summation problem can be scheduled so that the parallel complexity achieves the minimum of $f_{(+)} \cdot \lceil \log_2 N \rceil$ by using [10] $\lfloor N/2 \rfloor$ identical processors. This performance is guaranteed, modulo constants, for any positive N if the parallel algorithms remain computation bound. But the radius of a $\lfloor N/2 \rfloor$ processor linear array is greater than or equal to $t \cdot \lfloor N/4 \rfloor$, where t is the distance between connected processors. Since the radius represents a lower bound on the communication cost for this problem, the algorithm cannot be computation bound if
>
> $$N > 4 \cdot \left[\frac{f_{(+)}}{t} \cdot \log_2 N + \left(\frac{f_{(+)}}{t} + 1 \right) \right] \quad .$$
>
> For the parallel algorithm to be computation bound for arbitrary N, the radius of the $N/2$ processors must be bounded by $f_{(+)} \cdot \lceil \log_2 N \rceil$ for all N. No d dimensional array multiprocessor will satisfy this condition for large N if t and $f_{(+)}$ are bounded away from zero.

1.2.2 Lower Bounds

A serial algorithm describes a mapping from a set of data to a set of solution values specified by the problem. Define a *nontrivial* solution value to be one that is the result of a binary floating point operation. Define a set of solution values to be *independent* if no two of them have the same absolute values for all possible data. Let the set of nontrivial

---

[10] $\lfloor x \rfloor$ represents the largest integer less than or equal to x.

independent solution values of an algorithm a be

$$U = \{u_j \,|\, j = 1, \ldots, N_{a,u}\} \quad,$$

where $N_{a,u}$ is the number of these solution values. For this set to be independent, at least $N_{a,u}$ binary floating point operations are required to generate its values. This is a consequence of our assumption that unary negation is the only unary floating point operation.

Let the data that is required to produce these solution values be

$$G = \{g_k \,|\, k = 1, \ldots, N_{a,g}\} \quad.$$

That is, there is no g_k in this set that can be arbitrarily changed without changing at least one of the solution values. $N_{a,g}$ is the number of these data. For a datum to be required by a nontrivial solution value, some unary function of its value must be an operand of a binary floating point operation. Therefore, at least $\lceil N_{a,g}/2 \rceil$ binary floating point operations are required to use all of this data. Since addition is the least expensive binary floating point operation, we have proven the following lemma.

Lemma 1.1 *A lower bound on the serial complexity of an algorithm a is*

$$f_{(+)} \cdot \max\left\{ N_{a,u}, \left\lceil \frac{N_{a,g}}{2} \right\rceil \right\} \quad.$$

For $u_j \in U$, define $N_a(u_j)$ to be the amount of data that is required by a to compute u_j. Again, this means that there is no g_k in this set that can be arbitrarily changed without changing the value of u_j. Each datum must be the operand of a binary floating point operation, and this operation produces a result that will itself be the operand of a binary floating point operation. Summing the number of operations indicated by this argument leads to the result that the serial complexity of calculating u_j is bounded from below by $f_{(+)} \cdot (N_a(u_j) - 1)$.[11] The next lemma is a direct consequence of this.

---

[11] See Lemma 1 in Kuck [Kuc78, page 95].

Lemma 1.2 *A lower bound on the serial complexity of an algorithm a is*

$$f_{(+)} \cdot \max_{u_j \in U} (N_a(u_j) - 1) \quad .$$

For the rest of this section, assume that we know a lower bound on the serial complexity of the form $f_{(+)} \cdot C_s(N_a(u_j))$ for all $u_j \in U$, where $C_s(N) \geq N - 1$.

For a parallel implementation of a, say that a processor *collaborates* in the computation of u_j if changing the results of all of the floating point operations calculated by that processor can change the value of u_j. Define $P_a(u_j)$ to be the number of processors that collaborate in the calculation of u_j for a given parallel implementation of a. For a given multiprocessor, consider the subset of $P_a(u_j)$ processors with minimum radius. Let $r(P_a(u_j))$ be this radius. Then the following lemma is a consequence of the discussion in the example on page 4.

Lemma 1.3 *For all $u_j \in U$, the communication cost of a parallel implementation of an algorithm a on a given multiprocessor is bounded from below by $r(P_a(u_j))$.*

This follows directly from the fact that information is needed from all $P_a(u_j)$ processors in order to calculate u_j.

The parallel complexity of calculating u_j on a multiprocessor can be no faster than a parallel implementation of summing its required data. The data can only be reduced by a sequence of binary operations, and binary add is assumed to be the fastest nonunary floating point operation. Thus, one lower bound on the parallel compelxity is $f_{(+)} \cdot \lceil \log_2 N_a(u_j) \rceil$, as in the example on page 10. This fact and inequality 1.1 on page 9 prove the following lemma.

Lemma 1.4 *Assume that an algorithm a calculates u_j. Then the parallel complexity of a parallel implementation of a on a multiprocessor with identical processors is bounded from below by*

$$f_{(+)} \cdot \max \left\{ \left\lceil \frac{C_s(N_a(u_j))}{P_a(u_j)} \right\rceil , \lceil \log_2 N_a(u_j) \rceil \right\} \quad .$$

Define

$$N_{a,*} = \max_{u_j \in U} N_a(u_j)$$

and

$$P_{a,*} = \max_{u_j \in U} P_a(u_j) \quad .$$

Theorem 1.1 *The parallel cost of a parallel implementation of an algorithm a on a multiprocessor with identical multiprocessors is bounded from below by*

$$\max \left\{ r(P_{a,*}), f_{(+)} \cdot \left\lceil \frac{C_s(N_{a,*})}{P_{a,*}} \right\rceil, f_{(+)} \cdot \lceil \log_2 N_{a,*} \rceil \right\} \quad .$$

Proof: The total cost is bounded from below by $\max\{C_p, W_p\}$. The result then follows from Lemmas 1.3 and 1.4. ∎

Let $r(p)$ be the radius of the p processor subset of a multiprocessor that has the minimum radius.

Corollary 1.2 *For a P processor multiprocessor with identical processors, a lower bound on the parallel cost of any parallel implementation of a serial algorithm a is*

$$\min_{p \in \{1,...,P\}} \max \left\{ r(p), f_{(+)} \cdot \left\lceil \frac{C_s(N_{a,*})}{p} \right\rceil, f_{(+)} \cdot \lceil \log_2 N_{a,*} \rceil, f_{(+)} \cdot \left\lceil \frac{N_{a,u}}{P} \right\rceil, f_{(+)} \cdot \left\lceil \frac{N_{a,g}}{2P} \right\rceil \right\} \quad .$$

Proof: For any parallel implementation of an algorithm a on this multiprocessor, $P_{a,*}$ is a member of the set $\{1, \ldots, P\}$. Thus, the first three terms inside the max operator follow from Theorem 1.1. The other two terms come from Lemma 1.1 and inequality 1.1.

1.3 Numerical Algorithms for PDEs

In Sections 1.3.1 through 1.3.4 we describe the class of linear PDEs whose solutions we will be approximating and introduce some useful notation. In Section 1.3.5 we give a simple example of such a PDE. In Section 1.3.6 we describe the type of numerical approximations we are analyzing. We finish with a description of algorithmic features of

these numerical approximations. Some of these assumptions are simplistic, but the results derived in Chapters 2 and 3 carry over immediately for more realistic assumptions. This is discussed briefly in Chapters 4 and 5.

1.3.1 Problem Description

Let $\Re^d$ be the d dimensional Euclidean vector space. Let Ω be a compact subset of $\Re^d$. For any nonnegative integer k, let I_k be the k dimensional unit cube

$$I_k = [0,1] \times \cdots \times [0,1] \subset \Re^k \ .$$

1) *We assume that we are approximating the solution of a linear scalar partial differential equation defined on Ω whose solution operator can be represented by an expression of the form*

$$u(\bar{z}) = \sum_{i=1}^{\bar{l}} \int_{I_{d_i}} \Psi_i(\bar{z}, \bar{x}) g_i(\bar{x}) \, d\bar{x}$$

for any $\bar{z} \in \Omega$. $u(\bar{z})$ is the solution function, $\bar{l}$ is a positive integer, the set $\{d_i \,|\, i \in \{1, \ldots, \bar{l}\}\}$ is made up of nonnegative integers, and the set of functions $\{g_i \,|\, i \in \{1, \ldots, \bar{l}\}\}$ represent the problem data.

Thus, the solution function is the sum of $\bar{l}$ *components* of the form

$$u_i(\bar{z}) = \int_{I_{d_i}} \Psi_i(\bar{z}, \bar{x}) g_i(\bar{x}) \, d\bar{x} \ .$$

For the class of problems being considered here, the kernels $\{\Psi_i\}$ are linear functionals of the *Green's function* for the PDE. See Section 1.3.5, Chapter 4, and Butkovskiy [But82] for examples of this type of representation of the solution operator.

1.3.2 Kernel Assumptions

We assume the following properties about the functions $\{\Psi_i\}$ introduced in Section 1.3.1.

Notation

For a given k, let $B(\bar{x}; \delta)$ be the k dimensional open ball of radius δ centered on $\bar{x}$. Define the boundary of an arbitrary set X in $\Re^k$ to be the set

$$\left\{ \bar{x} \mid \forall \delta > 0 \ \left(B(\bar{x}; \delta) \cap X \neq \emptyset \ \text{ and } \ B(\bar{x}; \delta) \cap \Re^k - X \neq \emptyset \right) \right\} \ .$$

That is, no ball centered on a point in the boundary does not contain points in both X and $\Re^k - X$. Denote the boundary by ∂X. Define a function $\Psi(\bar{x})$ to be a type 1 function in the domain I_k if it satisfies the following 4 conditions.

a) $\Psi(\bar{x}) \in L^1(I_k)$, i.e. $\int_{I_k} |\Psi(\bar{x})| \, d\bar{x}$ is well defined and finite.

b) The boundary of the subset of I_k where $\Psi(\bar{x})$ is zero has measure zero in $\Re^k$.

c) $\Psi(\bar{x})$ is continuous everywhere in I_k except on a set that has measure zero in $\Re^k$.

d) If R is a set of measure zero in I_k, then $\int_R |\Psi(\bar{x})| \, d\bar{x} = 0$.

Assumptions

2) *For each $i \in \{1, \ldots, \bar{l}\}$ and $\bar{z} \in \Omega$, $\Psi_i(\bar{z}, \bar{x})$ is a type 1 function in I_{d_i}. Furthermore, for each $i \in \{1, \ldots, \bar{l}\}$,*

$$\max_{\bar{z} \in \Omega} \int_{I_{d_i}} |\Psi_i(\bar{z}, \bar{x})| \, d\bar{x}$$

is finite.

1.3.3   Data Assumptions

We assume the following conditions on the data functions $\{g_i\}$.

Notation

Let $C^m(I_{d_i})$ be the set of all functions that have continuous mth order partial derivatives on the set I_{d_i}. Let $K_{d_i}(m)$ be a set of d_i-vectors whose components are nonnegative

integers that sum to m, i.e.

$$K_{d_i}(m) = \left\{ \bar{\mu} \,\middle|\, \bar{\mu} = (\mu_1, \ldots, \mu_{d_i}) \text{ and } \sum_{j=1}^{d_i} \mu_j = m \right\} \quad .$$

Let $\bar{x} = (x_1, \ldots, x_{d_i})$. Let $\bar{x}^{\bar{\mu}} = x_1^{\mu_1} x_2^{\mu_2} \cdots x_{d_i}^{\mu_{d_i}}$. Represent a particular mth order partial derivative of a function $g \in C^m(\Re^{d_i})$ by[12]

$$g^{(\bar{\mu})}(\bar{x}) \;=\; \frac{\partial^{\bar{\mu}}}{\partial \bar{x}^{\bar{\mu}}} g(\bar{x}) \;=\; \frac{\partial^{\mu_1}}{\partial x_1^{\mu_1}} \cdots \frac{\partial^{\mu_{d_i}}}{\partial x_{d_i}^{\mu_{d_i}}} g(\bar{x}) \quad ,$$

where $\bar{\mu} \in K_{d_i}(m)$.

Let ∇_{d_i} be the gradient operator for $\Re^{d_i}$. Thus,

$$\nabla_{d_i} = \left(\frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_{d_i}} \right) \quad .$$

Define $\nabla_{d_i}^{(m)}$ recursively by

$$\nabla_{d_i}^{(m)} = \nabla_{d_i}^{(m-1)} \otimes \nabla_{d_i} \quad ,$$

where $\otimes$ is the tensor product [Hal74]. If $g \in C^m(\Re^{d_i})$, then the elements of $\nabla_{d_i}^{(m)} g(\bar{x})$ are the mth order partial derivatives of g at $\bar{x}$. Let $\bar{\nabla}_{d_i}^{(m)} g$ be a vector whose elements are some ordering of the elements of $\nabla_{d_i}^{(m)} g$.

Assumptions

3) *For each $i \in \{1, \ldots, \bar{l}\}$, we assume that g_i is known to be some member of a set G_i defined in the following way. G_i is the set of all functions g satisfying the properties*

   i) $g(\bar{x}) \in C^{m_i}(I_{d_i})$

   ii) $\|\bar{\nabla}_{d_i}^{(m_i)} g(\bar{x})\|_{(i)} \le M_i(\bar{x})$ *for all $\bar{x} \in I_{d_i}$*

   where m_i is a positive integer, $\|\cdot\|_{(i)}$ is a vector norm, and $M_i(\bar{x})$ is a bounded, nonnegative, type 1 function.

The simplest example of a bound on the norm is a *uniform bound.*

---

[12]This is the multiindex notation of Schwartz. See John [Joh82].

Example: A uniform bound on the m_ith order partial derivatives of g_i has the form

$$\|\bar{\nabla}_{d_i}^{(m_i)} g_i(\tilde{x})\|_{(i)} \le B \qquad \forall \tilde{x} \in I_{d_i} \quad,$$

where the norm is a vector norm for vectors of the appropriate length and B is a nonnegative real number. For example, if $d_i = 2$ and $m_i = 2$, then a global bound based on the discrete infinity norm is

$$\max\left\{\left|\frac{\partial^2}{\partial x_1^2}g_i(\tilde{x})\right|, \left|\frac{\partial^2}{\partial x_1 x_2}g_i(\tilde{x})\right|, \left|\frac{\partial^2}{\partial x_2^2}g_i(\tilde{x})\right|\right\} \le B$$

for all $\tilde{x} \in I_{d_i}$.

1.3.4   Compatibility Conditions

The PDE will also be well-defined for data functions other than the given ones. We assume the following sufficient conditions on permissible data functions for the PDE.

4) For each $i \in \{1, \ldots, \bar{l}\}$, we assume that any member of G_i is a permissible data function for the ith component of a solution to the PDE. We also assume that any combination of data functions from the sets $\{G_i\}$ generate a possible solution to the PDE, with one possible exception. The inclusion of a given data function g_i in a set of data functions may force the data function for a different component, g_j, to have given function and derivative values on the boundary of I_{d_j}. We will refer to this as a compatibility condition.

Each set of compatible data functions drawn from the sets $\{G_i\}$ define a solution function u. We will refer to the set of all solution functions generated in this fashion by U.

1.3.5   Example

Let

$$\Delta_{x_1, \ldots, x_d} = \sum_{j=1}^{d} \frac{\partial^2}{\partial x_j^2} \quad.$$

Consider the following elliptic PDE in two space dimensions,

$$-\Delta_{x_1,x_2} u = f(\bar{x}) \quad \text{on} \quad B((0,0);1)$$
$$u(\bar{x}) = h(\bar{x}) \quad \text{on} \quad \partial B((0,0);1) \ .$$

Then

$$u(\bar{z}) = \int_{B((0,0);1)} G(\bar{z},\bar{x}) f(\bar{x}) \, d\bar{x} \ + \int_{\partial B((0,0);1)} \left(\frac{\partial}{\partial n} G(\bar{z},\bar{x}) \right) h(\bar{x}) \, ds \ ,$$

where $\partial G(\bar{z},\bar{x})/\partial n$ is the normal derivative of G with respect to $\bar{x}$ on the surface of the unit ball. The Green's function $G(\bar{z},\bar{x})$ has a logarithmic singularity at $\bar{x} = \bar{z}$, and is a C^∞ function elsewhere in $B((0,0);1)$. See Butkovskiy [But82, pages 131–132] for more details.

To put this operator into the desired form, we first define the function $G(\bar{z},\bar{x})$ to be identically zero for $\bar{x} \in ([-1,1] \times [-1,1]) - B((0,0);1)$. The extension of the data function f into $([-1,1] \times [-1,1]) - B((0,0);1)$ can be done arbitrarily. As will become clear in the next chapter, it is best for the analysis if the extension is as smooth as possible. Note that I_2 is mapped onto the region $[-1,1] \times [-1,1]$ by the function

$$\varphi_1(x_1,x_2) = (2 \cdot x_1 - 1, 2 \cdot x_2 - 1) \ .$$

$\partial B((0,0);1)$ can be divided into two submanifolds,

$$\Upsilon_2 = \{(x_1,x_2) \mid x_1 \leq 0, x_1^2 + x_2^2 = 0\}$$

and

$$\Upsilon_3 = \{(x_1,x_2) \mid x_1 \geq 0, x_1^2 + x_2^2 = 0\} \ .$$

The unit interval is mapped onto Υ_2 by the function

$$\varphi_2(x) = \left(\cos(\frac{\pi}{2} + \pi \cdot x)), -\sin(\frac{\pi}{2} + \pi \cdot x)) \right) \ .$$

Similarly, the unit interval is mapped onto Υ_3 by the function

$$\varphi_3(x) = \left(\cos(\frac{\pi}{2} - \pi \cdot x)), \sin(\frac{\pi}{2} - \pi \cdot x)) \right) \ .$$

Therefore, the solution operator can be rewritten in the form

$$u(\bar{z}) \;=\; \int_{I_2} G(\bar{z},\varphi_1(\bar{x}))f(\varphi_1(\bar{x})) \cdot 4\, d\bar{x} \;+\; \int_0^1 \left(\frac{\partial}{\partial n} G(\bar{z},\varphi_2(x)) \right) h(\varphi_2(x)) \cdot \pi\, dx$$
$$+\; \int_0^1 \left(\frac{\partial}{\partial n} G(\bar{z},\varphi_3(x)) \right) h(\varphi_3(x)) \cdot \pi\, dx \;\; .$$

By the notation of Section 1.3.1,

$$\Psi_1(\bar{z},\bar{x}) \;=\; 4 \cdot G(\bar{z},\varphi_1(\bar{x})) \;\; ,$$
$$\Psi_2(\bar{z},x) \;=\; \pi \cdot \left(\frac{\partial}{\partial n} G(\bar{z},\varphi_2(x)) \right) \;\; ,$$
$$\Psi_3(\bar{z},x) \;=\; \pi \cdot \left(\frac{\partial}{\partial n} G(\bar{z},\varphi_3(x)) \right) \;\; ,$$

$$g_1(\bar{x}) = f(\bar{x}), \quad g_2(x) = h(\varphi_2(x)), \quad \text{and} \quad g_3(x) = h(\varphi_3(x)) \;\; .$$

If $\bar{z} \in B((0,0);1)$, then it is clear that each Ψ_i is a type 1 function in its domain. Note that g_2 and g_3 must satisfy the compatibility conditions that

$$g_2(0) = g_3(1)$$

and

$$g_2(1) = g_3(0) \;\; .$$

If h is known to be differentiable, then compatibility conditions will also exist on the value of the derivatives of g_2 and g_3 at $x = 0$ and $x = 1$.

1.3.6   Finite Approximations

Numerical approximations to the solution of the PDE replace the possibly infinite dimensional problem with a finite dimensional problem. The dimensionality of the problem is reduced by introducing error in the following sense:

- Only a finite amount of information about the solution function is calculated. Any model of the solution based on only this information will only approximate the true solution.

- Only a finite amount of information about the data functions is used to calculate the desired solution characteristics. We will refer to this as the *data function sampling*. Unless this information completely characterizes the data functions, the solution values that are calculated are also approximate.

The PDE is replaced by a relationship between the chosen information. The error in the approximation to the solution is also a function of what this relationship is.

For the rest of this dissertation, we restrict ourselves to the case where

5) *values of the data functions at given locations in their domains are used,*

6) *values of the solution function at given locations in its domain are approximated,*

7) *the error in approximating each solution value is bounded by some given value* ϵ.

We assume that an algorithm a calculates the value of the solution function u at some set of locations, $Z = \{\bar{z}_j \,|\, j = 1, \ldots, N_{a,u}\}$. And, for each data function g_i, the algorithm uses function values at some set of locations $X_i = \{\bar{x}_{i,k} \,|\, k = 1, \ldots, N_{a,i}\}$. For any particular solution value $u(\bar{z}_j)$, the algorithm will use values of g_i at some set of locations

$$X_{i,j} = \{\bar{x}_{i,j,k} \,|\, k = 1, \ldots, N_{a,i}(\bar{z}_j)\} \subseteq X_i$$

in I_{d_i}. Note the slight change in notation from Section 1.2.2. Instead of $N_{a,i}(u(\bar{z}_j))$, we use $N_{a,i}(\bar{z}_j)$.

Intrinsic errors

Let u be some solution function in U, and let $\{g_i\}$ be the corresponding data functions for u. Let $u_i(\bar{z}_j) = \int_{I_{d_i}} \Psi_i(\bar{z}_j, \bar{x}) g_i(\bar{x}) \, d\bar{x}$. For a given i, let $G_{a,i,j}(g_i)$ be the set of all data functions in G_i satisfying the following properties.

- If $g \in G_{a,i,j}(g_i)$, then g has the same values as g_i at the locations $\{\bar{x}_{i,j,k}\}$.

- If $g \in G_{a,i,j}(g_i)$ and $l \in \{0, \ldots, m_i\}$, then

$$\frac{\partial^{\bar{\mu}} g(\bar{x})}{\partial \bar{x}^{\bar{\mu}}} = \frac{\partial^{\bar{\mu}} g_i(\bar{x})}{\partial \bar{x}^{\bar{\mu}}}$$

for all $\bar{x} \in \partial I_{d_i}$ and all $\bar{\mu} \in K_{d_i}(l)$.

That is, if $g \in G_{a,i,j}(g_i)$, then g and g_i have the same values at all sampling locations, and g has the same function and derivative values as g_i on the boundary of I_{d_i}. Since compatiblity conditions only constrain functions on the boundary of their domain, and since g_i is necessarily compatible with the other data functions, each function in $G_{a,i,j}(g_i)$ is also compatible with the other data functions.

If $g_{i,1}, g_{i,2} \in G_{a,i,j}(g_i)$, then the difference between the solution values associated with these two data functions is

$$\int_{I_{d_i}} \Psi_i(\bar{z}_j, \bar{x})(g_{i,1}(\bar{x}) - g_{i,2}(\bar{x}))\, d\bar{x} \quad .$$

It is impossible to distinguish g_i from the other functions in $G_{a,i,j}(g_i)$ merely given the information $\{g_i(\bar{x}_{i,j,k})\}$. Call the uncertainty introduced by the data function sampling the *data sampling error*. Let $U_{a,i,j}(g_i)$ be the set of values generated by these functions,

$$U_{a,i,j}(g_i) = \{v_i \mid v_i = \int_{I_{d_i}} \Psi_i(\bar{z}_j, \bar{x})g(\bar{x})\, d\bar{x} \quad \text{for} \quad g \in G_{a,i,j}(g_i)\} \quad . \tag{1.2}$$

Let $D_{a,i,j}(g_i)$ be the maximum difference between these values,

$$\begin{aligned}
D_{a,i,j}(g_i) &= \max_{v_{i,1}, v_{i,2} \in U_{a,i,j}(g_i)} |v_1 - v_2| \\
&= \max_{g_{i,1}, g_{i,2} \in G_{a,i,j}(g_i)} \left| \int_{I_{d_i}} \Psi_i(\bar{z}_j, \bar{x})(g_{i,1}(\bar{x}) - g_{i,2}(\bar{x}))\, d\bar{x} \right| \quad .
\end{aligned}$$

Define the minimum worst case data sampling error for the ith component at $\bar{z}_j$ to be the tightest possible bound on the data sampling error,

$$\min_{g_{i,1} \in G_{a,i,j}(g_i)} \max_{g_{i,2} \in G_{a,i,j}(g_i)} \left| \int_{I_{d_i}} \Psi_i(\bar{z}_j, \bar{x})(g_{i,1}(\bar{x}) - g_{i,2}(\bar{x}))\, d\bar{x} \right| \quad .$$

This represents the amount of uncertainty in $u(\bar{z}_j)$ when only the ith component is being approximated.

Lemma 1.5 *The minimum worst case data sampling error for the ith component at $\bar{z}_j$ is greater than or equal to*

$$\frac{D_{a,i,j}(g_i)}{2} \quad .$$

Proof: Assume that there exists a $g_{i,*} \in G_{a,i,j}(g_i)$ such that

$$\max_{g \in G_{a,i,j}(g_i)} \left| \int_{I_{d_i}} \Psi_i(\bar{z}_j, \bar{x})(g_{i,*}(\bar{x}) - g(\bar{x})) \, d\bar{x} \right| < \frac{D_{a,i,j}(g_i)}{2} \; .$$

Then

$$\max_{g_{i,1}, g_{i,2} \in G_{a,i,j}(g_i)} \left| \int_{I_{d_i}} \Psi_i(\bar{z}_j, \bar{x})(g_{i,1}(\bar{x}) - g_{i,2}(\bar{x})) \, d\bar{x} \right|$$

$$\leq \max_{g \in G_{a,i,j}(g_i)} \left| \int_{I_{d_i}} \Psi_i(\bar{z}_j, \bar{x})(g(\bar{x}) - g_{1,*}(\bar{x})) \, d\bar{x} \right|$$

$$+ \max_{g \in G_{a,i,j}(g_i)} \left| \int_{I_{d_i}} \Psi_i(\bar{z}_j, \bar{x})(g_{i,*}(\bar{x}) - g(\bar{x})) \, d\bar{x} \right|$$

$$< \frac{D_{a,i,j}(g_i)}{2} + \frac{D_{a,i,j}(g_i)}{2}$$

$$= D_{a,i,j}(g_i) \; .$$

This is a contradiction. Therefore,

$$\max_{g \in G_{a,i,j}(g_i)} \left| \int_{I_{d_i}} \Psi_i(\bar{z}_j, \bar{x})(g_{i,*}(\bar{x}) - g(\bar{x})) \, d\bar{x} \right| \geq \frac{D_{a,i,j}(g_i)}{2}$$

for all $g_{i,*} \in G_{a,i,j}(g_i)$, and

$$\min_{g_{i,1} \in G_{a,i,j}(g_i)} \max_{g_{i,2} \in G_{a,i,j}(g_i)} \left| \int_{I_{d_i}} \Psi_i(\bar{z}_j, \bar{x})(g_{i,1}(\bar{x}) - g_{i,2}(\bar{x})) \, d\bar{x} \right| \geq \frac{D_{a,i,j}(g_i)}{2} \; .$$

∎

We can similarly define a minimum worst case data sampling error for u at $\bar{z}_j$. Let $G'_{a,i,j}(g_i)$ be the set of all data functions in G_i that have the same values as g_i at the locations $\{\bar{x}_{i,j,k}\}$. Let $U'_{a,i,j}(g_i)$ be the set of ith component values generated by these functions. Let $U_{a,j}(u)$ be the set of all solution values whose data functions have the same value as those for u at the sampling locations,

$$U_{a,j}(u) = \left\{ v \, \middle| \, v = \sum_{i=0}^{I} v_i, \; v_i \in U'_{a,i,j}(g_i) \right\} \bigcap U \; . \tag{1.3}$$

Let $D_{a,j}(u)$ be the maximum difference between these values,

$$D_{a,j}(u) = \max_{v_1, v_2 \in U_{a,j}(u)} |v_1 - v_2| \; .$$

Then the minimum worst case data sampling error for u at $\bar{z}_j$ is

$$\min_{v_1 \in U_{a,j}(u)} \max_{v_2 \in G_{a,j}(u)} |v_1 - v_2| \quad .$$

Lemma 1.6 *The minimum worst case data sampling error for computing $u(\bar{z}_j)$ is greater than or equal to*

$$\sum_{i=0}^{l} \frac{D_{a,i,j}(g_i)}{2} \quad .$$

Proof: For each i, $G_{a,i,j}(g_i) \subset G'_{a,i,j}(g_i)$. Moreover, any combination of data functions from the sets $\{G_{a,i,j}(g_i)\}$ is compatible, since each function is compatible with the true data functions. Therefore, if a function v has the form

$$v = \sum_{i=0}^{l} v_i \qquad v_i \in U_{a,i,j}(g_i) \quad ,$$

then $v(\bar{z}_j) \in U_{a,j}(u)$. In consequence,

$$
\begin{aligned}
D_{a,j}(u) &= \max_{v_1,v_2 \in U_{a,j}(u)} |v_1 - v_2| \\
&= \sum_{i=0}^{l} \max_{v_{i,1},v_{i,2} \in U_{a,i,j}(u)} |v_{i,1} - v_{i,2}| \\
&\geq \sum_{i=0}^{l} D_{a,i,j}(g_i) \quad .
\end{aligned}
$$

The same argument used in Lemma 1.5 proves that

$$\min_{v_1 \in U_{a,j}(u)} \max_{v_2 \in U_{a,j}(u)} |v_1 - v_2| \geq \frac{D_{a,j}(u)}{2} \quad .$$

Therefore,

$$\min_{v_1 \in U_{a,j}(u)} \max_{v_2 \in G_{a,j}(u)} |v_1 - v_2| \geq \sum_{i=0}^{l} \frac{D_{a,i,j}(g_i)}{2} \quad .$$

∎

Since these minimum worst case sampling errors are only functions of the sampling locations and the PDE, we call them *intrinsic* errors. For arbitrary G_i, these lower bounds on the intrinsic errors can be arbitrarily large. For a priori estimates of the error, some assumptions about the data functions are necessary. Assumption 3 on page 19 is a common type of assumption for piecewise polynomial approximation based methods, like finite difference [RM67] [BL84] and finite element [SF73] methods.

Modelling errors

Most numerical methods introduce more than the minimum worst case error when approximating the solution values. Tighter bounds can be derived by examining what model a particular method uses to represent the data functions from the given values. For example, standard Galerkin methods [Fle84] [SF73] [BTW84] explicitly introduce a model for the solution and each data function, $\tilde{u}$ and $\{\tilde{g}_i\}$. Even when the solution is the only function being explicitly modelled in the algorithm, the error introduced by the model can often be interpreted as error imposed upon the data.

Each model of a data function introduces error into the approximate solution in a fashion similar to the intrinsic error. For simplicity, assume that exactly one data function model is used to approximate g_i in the calculation of a given solution value $u(\bar{z}_j)$. Call this model $\tilde{g}_{i,j}$. Then the minimum worst case error in computing the ith component at $\bar{z}_j$ is

$$\max_{g \in G_{a,i,j}(g_i)} \int_{I_{d_i}} \Psi_i(\bar{z}_j, \bar{x})(g(\bar{x}) - \tilde{g}_{i,j}(\bar{x}))\, d\bar{x} \quad .$$

To bound the modelling error for the data functions satisfying the assumption on page 19, any model $\tilde{g}_{i,j}$ must satisfy

$$\int_{I_{d_i}} \Psi_i(\bar{z}_j, \bar{x})(\tilde{g}_{i,j} - g)\, d\bar{x} = 0$$

for all polynomials g of order less than m_i. This is a simple variant of Theorem 3.1 in [TW80, page 31]. It is sufficient if the model $\tilde{g}_{i,j}$ is an m_ith order accurate approximation to the data, i.e. all polynomials of degree less than m_i are represented exactly.

1.3.7 Linear Algorithms and Serial Complexity

For an algorithm a, let $\tilde{g}_i$ be a vector made up of the values of g_i used by the algorithm. If a is linear, then the finite dimensional problem can be expressed by a matrix equation of the form

$$A\tilde{u} = \sum_{i=0}^{\bar{l}} R_i \tilde{g}_i \quad , \tag{1.4}$$

where $\tilde{u}$ is a vector containing the solution values to be approximated. If all elements of $\tilde{u}$ are nontrivial functions of the data, then A must be expressible as a square nonsingular

matrix. If A is an $N \times N$ matrix and R_i is an $N \times M_i$ matrix, then $N \geq N_{a,u}$ and $M_i = N_{a,i}$. The serial complexity of solving this matrix equation is a function of the properties of the matrices, but

$$f_{(/)} \cdot N \; + \; f_{(*)} \cdot \sum_{i=0}^{I} N_{a,i} \; + f_{(+)} \cdot \max_{j} \left(\sum_{i=0}^{I} N_{a,i}(\bar{z}_j) \; - \; 1 \right) \qquad (1.5)$$

is a lower bound. The first two terms follow from the representation of the matrix problem, and the last term follows from Lemma 1.2 on page 15. Here $f_{(*)}$, $f_{(/)}$, and $f_{(+)}$ are the times to execute a floating point multiplication, division, and addition respectively. Note that the second term is a lower bound on forming the right hand side, and the first term is a lower bound on solving the matrix equation. This expression assumes that the coefficients in R_i and A that are equal to 1 are either not known, or not taken advantage of.

Explicit system

If A is a diagonal matrix, then this is an *explicit* method. $N = N_{a,u}$ for an efficient serial algorithm. If we assume that $\tilde{u}$ is ordered so that $\tilde{u}_j$ is the approximation to $u(\bar{z}_j)$, then $N_{a,i}(\bar{z}_j)$ is the number of nonzero entries in the jth row of R_i. Each multiplication of a row of R_i by $\tilde{g}_i$ requires $N_{a,i}(\bar{z}_j)$ multiplications and $N_{a,i}(\bar{z}_j) - 1$ additions. Therefore, the complexity of the straightforward method for calculating the solution of the matrix equation is

$$\sum_{j=1}^{N_u} \left(f_{(/)} + \sum_{i=0}^{I} \left(f_{(*)} \cdot N_{a,i}(\bar{z}_j) + f_{(+)} \cdot (N_{a,i}(\bar{z}_j) - 1) \right) \right) \quad . \qquad (1.6)$$

Again, the possibility of nonzero entries being known to have the value of 1 is ignored. If the R_i matrices have special properties, then this serial complexity can be reduced. The classic example is the fast fourier transform method, which reduces the serial complexity of a matrix-vector multiplication for a particular dense matrix from $\Theta(N^2)$ multiplications and additions[13] to $\Theta(N \log N)$ multiplications and additions. See [AHU74] for a description

---

[13]For two real valued functions f and g, $f(x) = \Theta(g(x))$ if there exist positive constants c_1, c_2, and x_0 such that

$$c_1 |g(x)| \leq |f(x)| \leq c_2 |g(x)|$$

for all $x > x_0$ [HS78].

of this method.

Implicit system

If A has more than one nonzero element in at least one row, then this is an *implicit* method. $N_{a,i}(\bar{z}_j)$ may no longer simply be the number of nonzero entries in the jth row of R_i. For many PDEs, the best known serial algorithms are implicit. Some of the work in good serial algorithms for solving the matrix equation is common to the calculation of multiple solution values. The serial complexity of calculating a single solution value will generally be larger for an implicit method than for a good explicit method, but the shared work when calculating all of the solution values can make it the better algorithm.

For an implicit method, the serial complexity is a function of both evaluating the right hand side and solving an equation of the form $Ax = b$. A general matrix equation solver has a serial complexity of $\Theta(N^3)$ multiplications and additions. But matrix equations generated by piecewise polynomial models for elliptic PDEs can be solved using multigrid algorithms in $\Theta(N)$ multiplications and additions. Both methods have $\Theta(N)$ floating point divisions. Other solvers for the matrix equation have complexities in between these two extremes, and all depend on assumptions about the properties of A. We will use the expression

$$c \cdot N^\gamma$$

to represent a bound on the number of multiplications and additions for some as yet unspecified matrix equation solver. We will always assume that $\gamma \in [1.3]$.

Other costs

This description of the serial complexity has ignored the cost of computing the elements of the matrices A and $\{R_i\}$. This can be considerable, although it is usually a linear function of the number of nonzero elements. Since it can be computed a priori, we will continue to ignore it. It will not affect the lower bounds described in the later chapters. But it can influence the comparison of different methods.

Chapter 2

Information Requirements

Let Z be a finite set of locations in Ω. For a PDE satisfying assumptions 1-4 in Section 1.3, define A to be the class of serial algorithms that approximate the solution of the PDE and satisfy the following condition:

- Each algorithm is based on a finite approximation satisfying assumptions 5-7 on page 22.

Let $A_\epsilon(Z)$ be the subset of A satisfying the following additional condition:

- Z is a subset of the locations where the solution function is approximated. And, for all $u \in U$, the error in approximating $u(\bar{z})$ at each location $\bar{z} \in Z$ is bounded by ϵ.

As in Section 1.3.6, let $N_{a,i}(\bar{z})$ be the number of values of g_i that are used by an algorithm a when approximating $u(\bar{z})$. Let $N_{\epsilon,i}(\bar{z})$ be the minimum number of values of g_i required by algorithms in $A_\epsilon(\{z\})$,

$$N_{\epsilon,i}(\bar{z}) = \min_{a \in A_\epsilon(\{\bar{z}\})} N_{a,i}(\bar{z}) \ .$$

That is, all algorithms in A that approximate $u(\bar{z})$ and satisfy an error tolerance of ϵ must use at least $N_{\epsilon,i}(\bar{z})$ values of g_i.

Lemma 2.1 *Let $f_{(+)}$ be a lower bound on the execution time of a floating point addition in a given multiprocessor. Then the expression*

$$f_{(+)} \cdot \max_{\bar{z} \in Z} \left\lceil \log_2 \left(\sum_{i=0}^{I} N_{\epsilon,i}(\bar{z}) \right) \right\rceil$$

is a lower bound on the parallel complexity of any parallel implemetation of an algorithm $a \in A_\epsilon(Z)$ on that multiprocessor.

Proof: If $\bar{z} \in Z$, then $A_\epsilon(Z) \subset A_\epsilon(\{\bar{z}\})$. Therefore, $N_{a,i}(\bar{z}) \geq N_{\epsilon,i}(\bar{z})$ for all $\bar{z} \in Z$ when $a \in A_\epsilon(Z)$. From Lemma 1.4 on page 15, a lower bound on the parallel complexity of a parallel algorithm a is

$$f_{(+)} \cdot \max_{j \in \{1,\ldots,N_{a,u}\}} \left\lceil \log_2 \left(\sum_{i=0}^{\bar{l}} N_{a,i}(\bar{z}_j) \right) \right\rceil \quad .$$

Therefore, a lower bound on the parallel complexity of any parallel implementation of any algorithm $a \in A_\epsilon(Z)$ is

$$f_{(+)} \cdot \max_{\bar{z} \in Z} \left\lceil \log_2 \left(\sum_{i=0}^{\bar{l}} N_{\epsilon,i}(\bar{z}) \right) \right\rceil \quad .$$

∎

In this chapter we calculate upper and lower bounds on $N_{\epsilon,i}(\bar{z})$ of the form

$$C \cdot \left(\frac{1}{\epsilon} \right)^{\frac{d_i}{m_i}}$$

when $N_{\epsilon,i}(\bar{z}) \neq 0$ and $M_i(\bar{x}) > 0$ for all $\bar{x} \in I_{d_i}$. We then define optimal parallel algorithms for the set $A_\epsilon(Z)$, and use the bounds on $N_{\epsilon,i}(\bar{z})$ to prove the following lower bound on their parallel complexity. If $N_{\epsilon,i}(\bar{z}) \geq C \cdot \epsilon^{-d_i/m_i}$ for even one $\bar{z} \in Z$ and one $i \in \{0,\ldots,\bar{l}\}$, then the parallel complexity of optimal parallel algorithms must increase linearly as a function of $d_i \cdot \log_2 \epsilon^{-1}$ when the error tolerance goes to zero. We also show that there is always a parallel algorithm based on an explicit linear algorithm whose parallel complexity is within a constant factor of this lower bound.

2.1 Lower Bound

Assume that $a \in A$. Let $u \in U$ be a solution function, and let $\{g_i\}$ be the corresponding set of data functions. As in Section 1.3.6, define $G_{a,i,j}(g_i)$ to be the set of all data

functions in G_i that have the same values as g_i at the locations $\{\bar{x}_{i,j,k}\}$, and have the same function and derivative values as g_i on the boundary of I_{d_i}. Define $D_{a,i,j}(g_i)$ by

$$D_{a,i,j}(g_i) = \max_{g_{i,1},g_{i,2} \in G_{a,i,j}(g_i)} \left| \int_{I_{d_i}} \Psi_i(\bar{z}_j, \bar{x})(g_{i,1}(\bar{x}) - g_{i,2}(\bar{x})) \, d\bar{x} \right| \quad .$$

Let $\bar{0}$ be the function that is identically zero on I_{d_i}. By our assumptions about the data functions, $\bar{0} \in G_i$. Let $A_{\epsilon,i}^0(\{\bar{z}_j\})$ be the subset of A that satisfies the following condition. If $a \in A_\epsilon^0(\{\bar{z}_j\})$, then $D_{a,i,j}(\bar{0}) \leq 2 \cdot \epsilon$. Define $N_{\epsilon,i}^0(\bar{z}_j)$ by

$$N_{\epsilon,i}^0(\bar{z}_j) = \min_{a \in A_{\epsilon,i}^0(\bar{z}_j)} N_{a,i}(\bar{z}_j) \quad .$$

Lemma 2.2 *For all $i \in \{0, \ldots, \bar{l}\}$,*

$$N_{\epsilon,i}(\bar{z}_j) \geq N_{\epsilon,i}^0(\bar{z}_j) \quad .$$

Proof: Let i' be some element of $\{0, \ldots, \bar{l}\}$. Assume that $a \in A$, but that $a \notin A_{\epsilon,i'}^0(\{\bar{z}_j\})$. Therefore, $D_{a,i',j}(\bar{0}) > 2 \cdot \epsilon$.

By the definition of U and the assumptions on $G_{i'}$, there exists some solution function $u \in U$ for which $g_{i'} = \bar{0}$. By Lemma 1.6, the minimum worst case data sampling error in approximating this solution function at $\bar{z}_j$ is bounded from below by

$$\frac{D_{a,i',j}(\bar{0})}{2} + \sum_{\substack{i=0 \\ i \neq i'}}^{l} \frac{D_{a,i,j}(g_i)}{2} \quad . \tag{2.1}$$

Since $D_{a,i,j}(g_i) \geq 0$ for all i, this bound is itself bounded from below by $D_{a,i',j}(\bar{0})/2$. Therefore, the lower bound on the error in expression 2.1 is strictly greater than ϵ for this solution function, and $a \notin A_\epsilon(\{\bar{z}_j\})$. Since a was an arbitrary algorithm not in $A_{\epsilon,i'}^0(\{\bar{z}_j\})$, this implies that $A_\epsilon(\{\bar{z}_j\}) \subset A_{\epsilon,i'}^0(\{\bar{z}_j\})$. It follows immediately that

$$N_{\epsilon,i'}(\bar{z}_j) \geq N_{\epsilon,i'}^0(\bar{z}_j) \quad .$$

Since i' was arbitrary, this proves the Lemma. ∎

In the rest of this section, we calculate lower bounds on the number of sampling locations necessary to satisfy

$$D_{a,i,j}(\bar{0}) \leq 2 \cdot \epsilon \quad .$$

The major tool in this analysis is the following lemma.

Lemma 2.3

$$D_{a,i,j}(\bar{0}) = 2 \cdot \max_{\gamma_i \in G_{a,i,j}(\bar{0})} \left| \int_{I_{d_i}} \Psi_i(\bar{z}_j, \bar{x}) \gamma_i(\bar{x}) \, d\bar{x} \right| \ .$$

Proof: If $\gamma_{i,*} \in G_{a,i,j}(\bar{0})$, then

$$\left\| \bar{\nabla}_{d_i}^{(m_i)}(-\gamma_{i,*}(\bar{x})) \right\|_{(i)} = \left\| \bar{\nabla}_{d_i}^{(m_i)} \gamma_{i,*}(\bar{x}) \right\|_{(i)} \leq M_i(\bar{x})$$

for all $\bar{x} \in I_{d_i}$. It is clear that $(-\gamma_{i,*})$ and its derivatives are zero at the same locations that $\gamma_{i,*}$ and its derivatives are zero. Therefore, $(-\gamma_{i,*}) \in G_{a,i,j}(\bar{0})$ and

$$\max_{\gamma_{i,1}, \gamma_{i,2} \in G_{a,i,j}(\bar{0})} \left| \int_{I_{d_i}} \Psi_i(\bar{z}_j, \bar{x})(\gamma_{i,1}(\bar{x}) - \gamma_{i,2}(\bar{x})) \, d\bar{x} \right|$$

$$\geq \max_{\gamma_{i,*} \in G_{a,i,j}(\bar{0})} \left| \int_{I_{d_i}} \Psi_i(\bar{z}_j, \bar{x}) \left(\gamma_{i,*}(\bar{x}) - (-\gamma_{i,*}(\bar{x})) \right) \, d\bar{x} \right|$$

$$= 2 \cdot \max_{\gamma_{i,*} \in G_{a,i,j}(\bar{0})} \left| \int_{I_{d_i}} \Psi_i(\bar{z}_j, \bar{x}) \gamma_{i,*}(\bar{x}) \, d\bar{x} \right| \ .$$

Also,

$$\max_{\gamma_{i,1}, \gamma_{i,2} \in G_{a,i,j}(\bar{0})} \left| \int_{I_{d_i}} \Psi_i(\bar{z}_j, \bar{x})(\gamma_{i,1}(\bar{x}) - \gamma_{i,2}(\bar{x})) \, d\bar{x} \right|$$

$$\leq \max_{\gamma_{i,1} \in G_{a,i,j}(\bar{0})} \left| \int_{I_{d_i}} \Psi_i(\bar{z}_j, \bar{x}) \gamma_{i,1}(\bar{x}) \, d\bar{x} \right| + \max_{\gamma_{i,2} \in G_{a,i,j}(\bar{0})} \left| \int_{I_{d_i}} \Psi_i(\bar{z}_j, \bar{x})(-\gamma_{i,2}(\bar{x})) \, d\bar{x} \right|$$

$$= 2 \cdot \max_{\gamma_{i,*} \in G_{a,i,j}(\bar{0})} \left| \int_{I_{d_i}} \Psi_i(\bar{z}_j, \bar{x}) \gamma_{i,*}(\bar{x}) \, d\bar{x} \right| \ .$$

∎

We will calculate a lower bound on $D_{a,i,j}(\bar{0})$ for a given set of data sampling locations by constructing a C^∞ function $\Gamma \in G_{a,i,j}(\bar{0})$. We will show that this lower bound on the intrinsic error is a function of m_i and the distance between the sampling locations. From this, we calculate a bound on the number of sampling locations we must use to satisfy a given error bound. To quantify the measure of distance, we introduce the Voronoi diagram of a set of sample locations.

2.1.1 Voronoi Diagram

We briefly describe Voronoi diagrams, and mention some of their properties. See also [GS85].

Discrete sampling locations

Consider a set of n distinct sampling locations $\{\bar{x}_k\}$ in $\Re^d$. The Voronoi diagram associated with these locations is a partition of $\Re^d$ made up of n open sets $\{V(\bar{x}_k)\}$ and one closed set, B. A point is in $V(\bar{x}_l)$ if it is closer to $\bar{x}_l$ than it is to any other sample location, i.e.

$$V(\bar{x}_l) = \{\bar{y} \mid \|\bar{x}_l - \bar{y}\|_2 < \|\bar{x}_k - \bar{y}\|_2, \ \forall k \neq l\} \ .$$

$\| \cdot \|_2$ is the Euclidean norm for $\Re^d$. We will call $V(\bar{x}_l)$ a Voronoi *cell*, and refer to $\bar{x}_l$ as the *center* of the cell. B is the complement of the union of the Voronoi cells. Every point on B satisfies the condition that the nearest sampling location is not unique. That is, each point is equidistant from at least two of the sample locations.

The Voronoi diagram can be constructed in the following fashion. Consider two locations in $\Re^d$, $\bar{x}_1$ and $\bar{x}_2$. Partition this space into three subregions, the $(d-1)$ dimensional hyperplane all of whose points are equidistant from both $\bar{x}_1$ and $\bar{x}_2$, and the two open half spaces defined by this hyperplane. Let $H_{1,2}(\bar{x}_1)$ and $H_{1,2}(\bar{x}_2)$ be the half spaces containing $\bar{x}_1$ and $\bar{x}_2$ respectively. For a set of n locations $\{\bar{x}_k\}$, consider a particular location $\bar{x}_l \in \{\bar{x}_k\}$. Identify the half spaces and bisecting hyperplanes constructed by a pairwise analysis of $\bar{x}_l$ and each of the other $n-1$ locations. Then $V(\bar{x}_l)$ is the intersection of all of those halfspaces that contain that location,

$$V(\bar{x}_l) = \bigcap_{k \neq l} H_{l,k}(\bar{x}_l) \ .$$

Thus, $V(\bar{x}_l)$ is a convex polygon. All other Voronoi cells are constructed in an analogous way. See Figure 2.1 for an example Voronoi diagram in $\Re^2$.

Smooth surfaces

We can also extend the concept of a Voronoi diagram to include a smooth surface. For example, let S be a $(d-1)$ dimensional differentiable surface in $\Re^d$, and let W be a set of
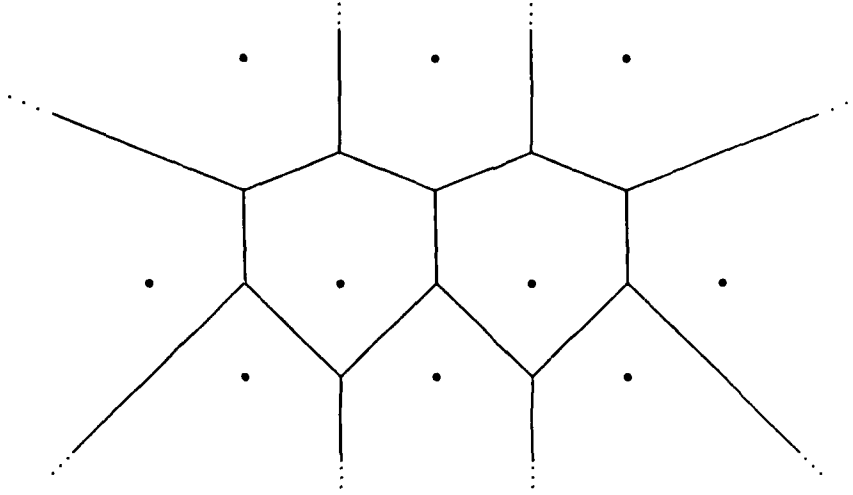
Figure 2.1: Example Voronoi diagram in $\Re^2$.

sampling locations bounded away from the surface. Treat every point on S as a sampling location. The Voronoi cell associated with a location $\bar{x}' \in S$ is again the set of points in $\Re^d$ that are closer to $\bar{x}'$ than to any other sampling location,

$$V(\bar{x}') = \{\bar{y} \mid \|\bar{x}' - \bar{y}\|_2 < \|\bar{x} - \bar{y}\|_2, \ \forall \bar{x} \in S \cup W, \ \bar{x} \neq \bar{x}'\} \ .$$

If $\bar{x}'$ is in the interior of S then $V(\bar{x}')$ is a segment of the line that is normal to S at $\bar{x}'$. The segment is only bounded if there is at least one sampling location in each half space defined by the plane tangent to S at $\bar{x}'$. $V(\bar{x}')$ is not an open set in $\Re^d$, but it is isomorphic to an open set in $\Re^1$. And B is still a closed set. The Voronoi cells for locations off the surface may no longer be polygons, but they are still convex. Note that this Voronoi diagram is a limiting case of the previous construction.

Sampling in a bounded region

For a final generalization, consider a finite discrete sampling $\{\bar{x}_k\}$ within a compact region $C \subset \Re^d$ whose boundary ∂C is the union of a finite number of smooth $(d-1)$ dimensional surfaces. Define the Voronoi diagram for these sample locations by first assuming that

every location on the boundary is also a sampling location, and then taking the intersection of the resulting Voronoi diagram with C. The Voronoi cells for the interior locations are naturally contained in C, so only the Voronoi cells with centers on the boundary are altered by being restricted to C.

Definitions and notation

Consider a finite set of sampling locations $\{\bar{x}_k\}$ in a compact region C whose boundary is the union of a finite number of smooth $(d-1)$ dimensional surfaces. Let $W = \{\bar{x}_k\} \bigcup \partial C$. Let $\bar{V}(\bar{w})$, $\bar{w} \in W$, be the closure of $V(\bar{w})$ in $\Re^d$. The *neighbors* of a cell $V(\bar{w}')$ are those cells whose closures intersect with the closure of $V(\bar{w}')$, i.e.

$$\{V(\bar{w}) \,|\, \bar{V}(\bar{w}') \cap \bar{V}(\bar{w}) \neq \emptyset \ , \bar{w} \in W \ , \ \bar{w} \neq \bar{w}'\} \ .$$

The *neighbors* of a sampling location $\bar{w}'$ are those sampling locations $\bar{w} \in W$ for which $V(\bar{w})$ is a neighbor of $V(\bar{w}')$.

Let the *boundary* of a cell $V(\bar{w}')$, $\partial V(\bar{w}')$, be the intersection of B with the closure of $V(\bar{w}')$. Every point on the boundary of one cell is also on the boundary of a neighboring cell. Thus, every location in the boundary of $V(\bar{w}')$ is equidistant from $\bar{w}'$ and the center of one of its neighbors, and is no closer to any other center. Let $r(\bar{w}')$ be the maximum distance between $\bar{w}'$ and the points on the boundary of $V(\bar{w}')$, i.e.

$$r(\bar{w}') = \max_{\bar{y} \in \partial V(\bar{w}')} \|\bar{w}' - \bar{y}\|_2 \ .$$

We will use this function as a measure of the distance between sample locations.

2.1.2 Error Function

In this section we construct a function $\Gamma \in G_{a,i,j}(\bar{0})$ using the Voronoi diagram corresponding to a set of data sampling locations.

Error function in I_{d_i}

Let $\bar{B}(\bar{x}; \delta)$ be the closure of $B(\bar{x}; \delta)$, the d_i dimensional ball centered on $\bar{x}$ with radius δ. Choose a location $\bar{x}_*$ in the interior of I_{d_i} and a $\delta > 0$ such that $\bar{B}(\bar{x}_*; \delta) \subset I_{d_i}$ and

$\Psi_i(\bar{z}_j, \bar{x})$ is continuous and nonzero on $\bar{B}(\bar{x}_*; \delta)$. Thus, since $\bar{B}(\bar{x}_*; \delta)$ is a compact set, $|\Psi_i(\bar{z}_j, \bar{x})|$ has a nonzero minima in $\bar{B}(\bar{x}_*, \delta)$.

If such a ball does not exist, then $u_i(\bar{z}_j) = 0$ and $N_{\epsilon,i}(\bar{z}_j) = 0$. This follows from Ψ_i being a type 1 kernel. The integral over the set of measure zero where Ψ_i is singular or discontinuous has no contribution, and Ψ_i is zero elsewhere. Thus, any error can be tolerated in the data function. For now, assume that $N_{\epsilon,i}(\bar{z}_j) \neq 0$.

Lemma 2.4 *If $d_i > 0$ and no sampling locations are located in $B(\bar{x}_*; \delta)$, then*

$$D_{a,i,j}(\bar{0}) \geq 2 \cdot C_i \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_*, \delta)} |\Psi_i(\bar{z}_j, \bar{x})| \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_*, \delta)} M_i(\bar{x}) \cdot \delta^{m_i + d_i} \quad ,$$

where C_i is a positive constant dependent only on m_i, d_i, and $\| \cdot \|_{(i)}$.

To prove this, we need the following fact.

Lemma 2.5 *Assume that $d_i > 0$. For any $\bar{x}_* \in I_{d_i}$ and $\delta > 0$ such that $B(\bar{x}_*; \delta) \subset I_{d_i}$, there exists a function $\gamma \in C^\infty(I_{d_i})$ with the following properties.*

1) $\gamma \in G_i$.

2) $\gamma(\bar{x}) = 0$ for all $\bar{x} \notin B(\bar{x}_*; \delta)$.

3) $\gamma(\bar{x}) \geq 0$ for all $\bar{x} \in B(\bar{x}_*; \delta)$.

4) There exists a constant $C_i > 0$ depending only on m_i, d_i, and $\| \cdot \|_{(i)}$ such that

$$\int_{I_{d_i}} \gamma(\bar{x}) \, d\bar{x} \geq C_i \cdot \delta^{m_i + d_i} \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_*; \delta)} M_i(\bar{x}) \quad .$$

The proof of Lemma 2.5 is a straightforward, but somewhat lengthy, construction. See Appendix A for the details. The proof of Lemma 2.4 follows directly from Lemma 2.5.

Proof of Lemma 2.4: Let $\gamma_{\bar{x}_*}$ be the function satisfying properties 1-4 of Lemma 2.5 on the ball $B(\bar{x}_*; \delta)$. Since there are no sampling locations in $B(\bar{x}_*; \delta)$, and since $\gamma_{\bar{x}_*}(\bar{x}) = 0$ for $\bar{x} \notin B(\bar{x}; \delta)$, $\gamma_{\bar{x}_*}$ is also a member of $G_{a,i,j}(\bar{0})$. Since $\gamma_{\bar{x}_*}(\bar{x})$ is nonnegative and $\Psi_i(\bar{z}_j, \bar{x})$ is of one sign in $B(\bar{x}_*; \delta)$,

$$\int_{\bar{B}(\bar{x}_*; \delta)} \Psi_i(\bar{z}_j, \bar{x}) \, \gamma_{\bar{x}_*}(\bar{x}) \, d\bar{x} = \pm \int_{\bar{B}(\bar{x}_*; \delta)} |\Psi_i(\bar{z}_j, \bar{x})| \, \gamma_{\bar{x}_*}(\bar{x}) \, d\bar{x} \quad .$$

Therefore, by Lemma 2.3,

$$
\begin{aligned}
D_{a,i,j}(\bar{0}) &\geq 2 \cdot \max_{\gamma \in G_{a,i,j}(\bar{0})} \left| \int_{I_{d_i}} \Psi_i(\bar{z}_j, \bar{x})\, \gamma(\bar{x})\, d\bar{x} \right| \\
&\geq 2 \cdot \left| \int_{B(\bar{x}_\bullet; \delta)} \Psi_i(\bar{z}_j, \bar{x})\, \gamma_{\bar{x}_\bullet}(\bar{x})\, d\bar{x} \right| \\
&\geq 2 \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_\bullet; \delta)} |\Psi_i(\bar{z}_j, \bar{x})| \cdot \int_{B(\bar{x}_\bullet; \delta)} \gamma_{\bar{x}_\bullet}(\bar{x})\, d\bar{x} \\
&\geq 2 \cdot C_i \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_\bullet; \delta)} |\Psi_i(\bar{z}_j, \bar{x})| \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_\bullet; \delta)} M_i(\bar{x}) \cdot \delta^{d_i + m_i} \quad .
\end{aligned}
$$

This proves the Lemma. ∎

To decrease the error below the bound in Lemma 2.4, the data function must be sampled in $B(\bar{x}_\bullet; \delta)$. Assume that $B(\bar{x}_\bullet; \delta)$ contains n data samples $\{\bar{x}_k\}$. As described in Section 2.1.1, these locations and the boundary of the ball define a Voronoi diagram in $\bar{B}(\bar{x}_\bullet; \delta)$. Let $W = \{\bar{x}_k\}$. Construct a function $\Gamma \in G_{a,i,j}(\bar{0})$ in the following way:

a) For each Voronoi cell $V(\bar{w})$, $\bar{w} \in W$, identify a location $\bar{y}_{\bar{w}}$ on its boundary where the function

$$
\|\bar{y} - \bar{w}\|_2 \quad \text{for} \quad \bar{y} \in \partial V(\bar{w})
$$

reaches its maximum, i.e.

$$
\|\bar{y}_{\bar{w}} - \bar{w}\|_2 = r(\bar{w}) \quad .
$$

b) Let $\tilde{W} = W$, $W' = \emptyset$, and $\Gamma_{\text{old}} = \bar{0}$.

c) Choose an element $\bar{w}' \in \tilde{W}$ that maximizes $r(\bar{w})$ over the set $\tilde{W}$. Since $\bar{y}_{\bar{w}'} \in \partial V(\bar{w}')$, it is no closer than $r(\bar{w}')$ to any of the other locations in $\tilde{W}$. Therefore $B(\bar{y}'_x; r(\bar{w}'))$ contains no sample locations. Let $\gamma_{\bar{w}'}$ be the function described in Lemma 2.5 for the ball $B(\bar{y}_{\bar{w}'}; r(\bar{w}'))$. Add $\bar{w}'$ to W', and set $\Gamma_{\text{new}} = \Gamma_{\text{old}} + \gamma_{\bar{w}'}$.

d) Let $S(\bar{w}')$ be the subset $\tilde{W}$ such that $\|\bar{y}_{\bar{w}'} - \bar{w}\| < 3r(\bar{w}')$ if $\bar{w} \in S(\bar{w}')$. Remove the locations in $S(\bar{w}')$ from $\tilde{W}$. Set Γ_{old} equal to Γ_{new}.

e) If $\tilde{W}$ is the empty set, then stop. Otherwise, return to step c.

Upon termination, $\tilde{W} = \emptyset$ and

$$\Gamma(\bar{x}) = \sum_{\bar{w}' \in W'} \gamma_{\bar{w}'}(\bar{x}) \ . \tag{2.2}$$

Let n' be the number of elements in W'. The following three lemmas verify that the construction is well-defined, and that Γ has the desired properties.

Lemma 2.6 *The construction of Γ is a well-defined process.*

Proof: For step a, we must be able to find a $\bar{y}_{\bar{w}}$ for any Voronoi cell $V(\bar{w})$. Since the function $\|\bar{y} - \bar{w}\|_2$ is continuous for $\bar{y} \in \partial V(\bar{w})$, and the boundaries of these Voronoi cells are closed bounded sets, there exists some $\bar{y}_{\bar{w}}$ such that $\|\bar{y}_{\bar{w}} - \bar{w}\|_2 = r(\bar{w})$ for each $\bar{w} \in W$.

For step c, we must be able to find a $\bar{w}' \in \tilde{W}$ that maximizes $r(\bar{w})$ over this set. There are never more than n sampling locations in $\tilde{W}$, so the cell that maximizes $r(\bar{w})$ in this set is well-defined.

Finally, step d always removes at least one element from $\tilde{W}$. Since it begins with n elements, the construction terminates after no more than n additions to Γ. ∎

Lemma 2.7 *The function Γ is the sum of a finite number of nonnegative $C^\infty(I_{d_i})$ functions $\{\gamma_k\}$ each of which satisfies the following three conditions.*

1) The support of γ_k is wholly contained in $B(\bar{x}_*; \delta)$.

2) $\gamma_k \in G_{a,i,j}(\bar{0})$.

3) The support of γ_k is disjoint from the support of the other functions in the set.

Proof: Γ is the sum of the n' functions $\{\gamma_{\bar{w}'} \mid \bar{w}' \in W'\}$. By the argument in Lemma 2.6, $n' < n$, and so is finite. From Lemma 2.5, we know that each $\gamma_{\bar{w}'}$ is a nonnegative $C^\infty(I_{d_i})$ function, and a member of G_i.

We also know that $\gamma_{\bar{w}'}(\bar{x}) = 0$ if $\bar{x} \notin B(\bar{y}_{\bar{w}'}; r(\bar{w}'))$ for each $\bar{w}' \in W'$. Since $\bar{y}_{\bar{w}'}$ is no closer than $r(\bar{w}')$ to any other Voronoi cell center, and since all locations in $\partial B(\bar{x}_*; \delta)$ are centers of Voronoi cells, $\bar{y}_{\bar{w}'}$ is no closer than $r(\bar{w}')$ to $\partial \bar{B}(\bar{x}_*; \delta)$. Therefore

$B(\bar{y}_{\bar{w}'}; r(\bar{w}')) \subset B(\bar{x}_*; \delta)$, and the support of each $\gamma_{\bar{w}'}$ is restricted to $B(\bar{y}_{\bar{w}'}; r(\bar{w}'))$. This proves the preamble and statement 1.

From step c of the construction, we know that $B(\bar{y}_{\bar{w}'}; r(\bar{w}'))$ contains no sample locations that are strictly inside $B(\bar{x}_*; \delta)$. And, from the previous paragraph, we know that $B(\bar{y}_{\bar{w}'}; r(\bar{w}'))$ contains no sample locations that are outside of $B(\bar{x}_*; \delta)$. Therefore, $\gamma_{\bar{w}'}$ is zero at all sampling locations. Also from the previous paragraph, we know that each $\gamma_{\bar{w}'}$ is identically zero outside of $B(\bar{x}_*; \delta)$. Since $\gamma_{\bar{w}'}$ is a C^∞ function, all derivatives of $\gamma_{\bar{w}'}$ are zero outside of $B(\bar{x}_*; \delta)$, and $\gamma_{\bar{w}'}$ satisfies any compatibility conditions associated with the data function $\bar{0}$. Therefore $\gamma_{\bar{w}'} \in G_{a,i,j}(\bar{0})$ for all $\bar{w}' \in W'$. This proves statement 2.

Assume that $\bar{y} \in I_{d_i}$ is in the support of both $\gamma_{\bar{w}'_j}$ and $\gamma_{\bar{w}'_k}$ for $\bar{w}'_j, \bar{w}'_k \in W'$. Since the support of each $\gamma_{\bar{w}'_j}$ is restricted to $B(\bar{y}_{\bar{w}'_j}; r(\bar{w}'_j))$,

$$\|\bar{y}_{\bar{w}'_j} - \bar{y}_{\bar{w}'_k}\|_2 < r(\bar{w}'_j) + r(\bar{w}'_k) \quad .$$

Therefore

$$\|\bar{w}'_j - \bar{y}_{\bar{w}'_k}\|_2 < 2r(\bar{w}'_j) + r(\bar{w}'_k)$$

and

$$\|\bar{w}'_k - \bar{y}_{\bar{w}'_j}\|_2 < r(\bar{w}'_j) + 2r(\bar{w}'_k) \quad .$$

Without loss of generality, assume that $\gamma_{\bar{w}'_j}$ was added to Γ first. Then $r(\bar{w}'_k) \le r(\bar{w}'_j)$, and $\|\bar{w}'_k - \bar{y}_{\bar{w}'_j}\|_2 < 3r(\bar{w}'_j)$. By step d of the construction, $\bar{w}'_k$ would have been removed from $\tilde{W}$ immediately after $\gamma_{\bar{w}'_j}$ was added, and thus would never have been used. This leads to a contradiction. Therefore we know that

$$\|\bar{y}_{\bar{w}'_j} - \bar{y}_{\bar{w}'_k}\|_2 > r(\bar{w}'_j) + r(\bar{w}'_k) \quad ,$$

and that $B(\bar{y}_{\bar{w}'_j}; \delta) \cap B(\bar{y}_{\bar{w}'_k}; \delta) = \emptyset$ for all $\bar{w}'_j, \bar{w}'_k \in W'$. This proves the last statement of the Lemma. ∎

Lemma 2.8 Γ *is a nonnegative* $C^\infty(I_{d_i})$ *function that is a member of* $G_{a,i,j}(\bar{0})$, *and whose support is wholly contained in* $B(\bar{x}_*; \delta)$.

Proof: By statements 1 and 2 of Lemma 2.7, Γ is the sum of a finite number of nonnegative C^∞ functions that are elements of $G_{a,i,j}(\bar{0})$, and that are identically zero outside

of $B(\bar{x}_*; \delta)$. Therefore Γ is also a nonnegative C^∞ function, $\Gamma(\bar{x}) = 0$ if $\bar{x} \in \{\bar{x}_k\}$, and $\Gamma(\bar{x}) = 0$ if $\bar{x} \notin B(\bar{x}_*; \delta)$. All that is left to prove is that

$$\|\bar{\nabla}_{d_i}^{(m_i)} \Gamma(\bar{x})\|_{(i)} \leq M_i(\bar{x})$$

for all $\bar{x} \in I_{d_i}$.

From equation 2.2,

$$\frac{\partial^{\bar{s}}}{\partial \bar{x}^{\bar{s}}} \Gamma(\bar{x}) = \sum_{\bar{w}' \in W'} \frac{\partial^{\bar{s}}}{\partial \bar{x}^{\bar{s}}} \gamma_{\bar{w}'}(\bar{x})$$

for any multiindex $\bar{s}$. By statement 3 of Lemma 2.7, the support of the functions $\{\gamma_{\bar{w}'}\}$ are pairwise disjoint, and

$$\frac{\partial^{\bar{\mu}}}{\partial \bar{x}^{\bar{\mu}}} \Gamma(\bar{x}) = \begin{cases} \frac{\partial^{\bar{\mu}}}{\partial \bar{x}^{\bar{\mu}}} \gamma_{\bar{w}'}(\bar{x}), & \text{for } \bar{x} \in B(\bar{y}_{\bar{w}'}; r(\bar{w}')), \ \bar{w}' \in W'; \\ 0, & \text{otherwise,} \end{cases}$$

for $\bar{\mu} \in K_{d_i}(m_i)$. Therefore,

$$\|\bar{\nabla}_{d_i}^{(m_i)} \Gamma(\bar{x})\|_{(i)} = \begin{cases} \|\bar{\nabla}_{d_i}^{(m_i)} \gamma_{\bar{w}'}(\bar{x})\|_{(i)}, & \text{for } \bar{x} \in B(\bar{y}_{\bar{w}'}; r(\bar{w}')), \ \bar{w}' \in W'; \\ 0 & \text{otherwise.} \end{cases}$$

Since $\gamma_{\bar{w}'} \in G_i$ for all $\bar{w}' \in W'$, $\|\bar{\nabla}_{d_i}^{(m_i)} \Gamma(\bar{x})\|_{(i)} \leq M_i(\bar{x})$ for all $\bar{x} \in I_{d_i}$. ∎

The next lemma implies that the support of Γ covers a significant portion of $B(\bar{x}_*; \delta)$. The proof is in AppendixB.

Lemma 2.9

$$B(\bar{x}_*; \delta) \subseteq \bigcup_{\bar{w}' \in W'} B(\bar{y}_{\bar{w}'}; 5r(\bar{w}')) \ .$$

The major result concerning this construction is the following theorem. It describes how the lower bound we have constructed can vary as the sample locations vary.

Theorem 2.1 *Assume that* $N_{\epsilon,i}(\bar{z}_j) \neq 0$. *If* $d_i = 0$, *then* $N_{\epsilon,i}(\bar{z}_j) = 1$. *If* $d_i > 0$, *let* $\bar{B}(\bar{x}_*; \delta) \subset I_{d_i}$ *be a closed ball in which* $\Psi_i(\bar{z}_j, \bar{x})$ *is nonzero and continuous as a function of* $\bar{x}$. *If* n *sample locations lie in the open ball* $B(\bar{x}_*; \delta)$, *then* $D_{a,i,j}(\bar{0})$ *is bounded from below by*

$$2 \cdot C_i \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_*, \delta)} |\Psi_i(\bar{z}_j, \bar{x})| \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_*, \delta)} M_i(\bar{x}) \cdot \begin{cases} \delta^{m_i + d_i}, & \text{if } n = 0; \\ \sum_{k=1}^{n} r_k^{m_i + d_i}, & \text{if } n > 0. \end{cases}$$

where

$$\sum_{k=1}^{n} r_k^{d_i} \geq \left(\frac{\delta}{5}\right)^{d_i} \quad if \ n > 0$$

and

$$0 \leq r_k \leq \delta \quad for \ 1 \leq k \leq n \ .$$

C_i *is the constant from Lemma 2.5 for a d_i dimensional domain.*

Proof: If $d_i = 0$, then I_{d_i} is a single point, and $N_{\epsilon,i}(\bar{z}_j)$ is either zero or one. Assume that $d \cdot > 0$.

If $n = 0$, then the Theorem follows directly from Lemma 2.4. If $n > 0$, then let $\{\bar{w}_k'\}$ be the sample locations used in the construction of Γ. Let n' be the number of sample locations used. By Lemmas 2.3, 2.5, 2.7 and 2.8, and equation 2.2,

$$D_{a,i,j}(\bar{0}) \geq 2 \cdot \max_{\gamma \in G_{a,i,j}(\bar{0})} \left| \int_{I_{d_i}} \Psi_i(\bar{z}_j, \bar{x}) \, \gamma(\bar{x}) \, d\bar{x} \right|$$

$$\geq 2 \cdot \left| \int_{B(\bar{x}_\bullet; \delta)} \Psi_i(\bar{z}_j, \bar{x}) \, \Gamma(\bar{x}) \, d\bar{x} \right|$$

$$\geq 2 \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_\bullet; \delta)} |\Psi_i(\bar{z}_j, \bar{x})| \cdot \left| \int_{B(\bar{x}_\bullet; \delta)} \Gamma(\bar{x}) \, d\bar{x} \right|$$

$$\geq 2 \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_\bullet; \delta)} |\Psi_i(\bar{z}_j, \bar{x})| \cdot \sum_{k=1}^{n'} \int_{B(\bar{y}_{\bar{w}_k'}; r(\bar{w}_k'))} \gamma_{\bar{w}_k'}(\bar{x}) \, d\bar{x}$$

$$\geq 2 \cdot C_i \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_k; \delta)} |\Psi_i(\bar{z}_j, \bar{x})| \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_\bullet; \delta)} M_i(\bar{x}) \cdot \sum_{k=1}^{n'} r(\bar{w}_k')^{d_i + m_i} \ .$$

By Lemma 2.9,

$$B(\bar{x}_\bullet; \delta) \subseteq \bigcup_{k=1}^{n'} B(\bar{y}_{\bar{w}_k'}; 5r(\bar{w}_k')) \ .$$

Therefore,

$$\mathsf{Vol}\,(B(\bar{x}_\bullet; \delta)) \leq \bigcup_{k=1}^{n'} \mathsf{Vol}\,\left(B(\bar{y}_{\bar{w}_k'}; 5r(\bar{w}_k'))\right) \ ,$$

or

$$\delta^{d_i} \leq \sum_{k=1}^{n'} (5r(\bar{w}_k'))^{d_i} \ .$$

Setting $r_k = r(\bar{w}_k')$ for $1 \leq k \leq n'$ and $r_k = 0$ for $k > n'$ produces the desired bound. This bound verifies the Theorem. ∎

2.1.3   Bound on $N_{\epsilon,i}(\bar{z}_j)$

Assume that an algorithm a uses a given number of sampling locations in I_{d_i}. Then $D_{a,i,j}(\bar{0})$ can be bounded from below by minimizing the error bound in Theorem 2.1. There will exist a minimum number of sample locations that guarantees that $D_{a,i,j}(\bar{0})$ is less than a given error tolerance. By Lemma 2.2, this number is a lower bound on $N_{\epsilon,i}(\bar{z}_j)$.

Lemma 2.10 *If $m_i \geq 1$ and $d_i \geq 1$, then the problem*

$$\min \sum_{k=1}^{n} r_k^{d_i + m_i}$$

subject to

$$\sum_{k=1}^{n} r_k^{d_i} \geq \left(\frac{\delta}{5}\right)^{d_i}$$

is solved when

$$r_k = \frac{\delta}{5} \cdot \left(\frac{1}{n}\right)^{\frac{1}{d}} \quad \forall k \in \{1, \ldots, n\} \ .$$

The minimum value of the objective function is

$$\left(\frac{1}{5}\right)^{d_i + m_i} \cdot \left(\frac{\delta}{n^{\frac{1}{d_i}}}\right)^{m_i} \cdot \delta^{d_i} \ .$$

Proof: Add the constraint that $\sum_{k=1}^{n} r_k^{d_i} = q$ for some given $q \geq (\delta/5)^{d_i}$. By LaGrange's method [Apo74] and inspection of the objective function, we know that the minimum of the augmented problem must satisfy

$$\frac{\partial}{\partial r_k} \left(\sum_{l=1}^{n} r_l^{d_i + m_i} + \lambda \cdot \sum_{l=1}^{n} r_l^{d_i} \right) = 0 \quad \forall k$$

$$\lambda \cdot \sum_{l=1}^{n} r_l^{d_i} = q \ .$$

The unique solution to this condition is

$$r_k = (q/n)^{\frac{1}{d_i}} \quad \forall k \ . \tag{2.3}$$

The corresponding value of the objective function is

$$q \cdot (q/n)^{\frac{m_i}{d_i}} \quad . \tag{2.4}$$

Expression 2.4 is minimized by choosing $q = (\delta/5)^{d_i}$.

For this value of q, expression 2.4 is a lower bound on the objective function for any feasible set of $\{r_k\}$. Since this lower bound is achieved for the r_k values described in equation 2.3, this must be the minimum value for the objective in the original minimization problem. ∎

The following lemma is a direct consequence of Lemma 2.10 and Theorem 2.1, where $B(\bar{x}_*; \delta)$ satisfies the usual assumptions.

Lemma 2.11 *If $N_{\epsilon,i}(\bar{z}_j) \neq 0$, $d_i > 0$, and $n > 0$, then*

$$2 \cdot C_i \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_*,\delta)} |\Psi_i(\bar{z}_j, \bar{x})| \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_*,\delta)} M_i(\bar{x}) \cdot \left(\frac{1}{5}\right)^{d_i + m_i} \cdot \left(\frac{\delta}{n^{\frac{1}{d_i}}}\right)^{m_i} \cdot \delta^{d_i}$$

is a lower bound on $D_{a,i,j}(\bar{0})$ for any algorithm $a \in A$ with n sampling locations in $B(\bar{x}_*; \delta)$.

Note that this lower bound comes from setting $r_k = \frac{1}{5} \cdot \delta \cdot n^{-1/d_i}$ for $k \in \{1, \ldots, n\}$. This is not achieveable using only n sample locations, but it is approximated by equidistributing the sample locations. In this case,

$$r_k \approx \frac{\delta}{n^{\frac{1}{d_i}}} \approx \frac{1}{h}$$

where h is the distance between sample locations.

Lemma 2.11 is sufficient to bound $N_{\epsilon,i}(\bar{z}_j)$.

Theorem 2.2 *Let $\bar{B}(\bar{x}_*; \delta) \subset I_{d_i}$ be any closed ball in which $\Psi_i(\bar{z}_j, \bar{x})$ is nonzero and continuous as a function of $\bar{x}$. If*

$$C_i \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_*,\delta)} |\Psi_i(\bar{z}_j, \bar{x})| \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_*,\delta)} M_i(\bar{x}) \cdot \delta^{m_i + d_i} > \epsilon \quad ,$$

then

$$N_{\epsilon,i}(\bar{z}_j) \geq \left\lceil \left(C_i' \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_*,\delta)} |\Psi_i(\bar{z}_j, \bar{x})| \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_*,\delta)} M_i(\bar{x}) \cdot \delta^{m_i + d_i} \right)^{\frac{d_i}{m_i}} \cdot \left(\frac{1}{\epsilon}\right)^{\frac{d_i}{m_i}} \right\rceil \quad .$$

where C_i' is a positive constant dependent only on m_i, d_i, and $\| \cdot \|_{(i)}$.

Proof: By Lemmas 2.4 and 2.2, if

$$C_i \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_\bullet, \delta)} |\Psi_i(\bar{z}_j, \bar{x})| \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_\bullet, \delta)} M_i(\bar{x}) \cdot \delta^{m_i + d_i} > \epsilon \ ,$$

then at least one sampling location is required in $B(\bar{x}; \delta)$ in order to satisfy the error tolerance. Let $C_i' = C_i \cdot (1/5)^{m_i + d_i}$. By Lemma 2.11, a necessary condition for an algorithm a to satisfy $D_{a,i,j}(\bar{0}) \leq 2 \cdot \epsilon$ is that

$$C_i' \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_\bullet, \delta)} |\Psi_i(\bar{z}_j, \bar{x})| \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_\bullet, \delta)} M_i(\bar{x}) \cdot \left(\frac{\delta}{n^{\frac{1}{d_i}}}\right)^{m_i} \cdot \delta^{d_i} \leq \epsilon \ ,$$

or

$$C_i' \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_\bullet, \delta)} |\Psi_i(\bar{z}_j, \bar{x})| \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_\bullet, \delta)} M_i(\bar{x}) \cdot \delta^{m_i + d_i} \cdot \left(\frac{1}{\epsilon}\right) \leq n^{\frac{m_i}{d_i}} \ , \qquad (2.5)$$

where n is the number of sample locations in $B(\bar{x}_\bullet; \delta)$. Since the total number of sample locations is greater than or equal to n, $a \notin A^0_{\epsilon,i}(\bar{z}_j)$ unless inequality 2.5 holds. Therefore,

$$N^0_{\epsilon,i}(\bar{z}_j) \geq \left\lceil \left(C_i' \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_\bullet, \delta)} |\Psi_i(\bar{z}_j, \bar{x})| \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_\bullet, \delta)} M_i(\bar{x}) \cdot \delta^{m_i + d_i}\right)^{\frac{d_i}{m_i}} \cdot \left(\frac{1}{\epsilon}\right)^{\frac{d_i}{m_i}} \right\rceil \ .$$

The proof of the Theorem follows immediately from Lemma 2.2. ∎

Thus, for most problems satisfying our assumptions, $N_{\epsilon,i}(\bar{z})$ grows at least linearly as a function of ϵ^{-d_i/m_i} when $\epsilon \to 0$. Note that the condition

$$C_i \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_\bullet, \delta)} |\Psi_i(\bar{z}_j, \bar{x})| \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_\bullet, \delta)} M_i(\bar{x}) \cdot \delta^{m_i + d_i} > \epsilon$$

always holds if

$$\left(C_i' \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_\bullet, \delta)} |\Psi_i(\bar{z}_j, \bar{x})| \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_\bullet, \delta)} M_i(\bar{x}) \cdot \delta^{m_i + d_i}\right)^{\frac{d_i}{m_i}} \cdot \left(\frac{1}{\epsilon}\right)^{\frac{d_i}{m_i}} > \left(\frac{1}{5}\right)^{\frac{d_i}{m_i(m_i + d_i)}} \ .$$

This follows from the fact that $C_i' = C_i \cdot 5^{-(m_i + d_i)}$. So the condition need only be checked if

$$\left(C_i' \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_\bullet, \delta)} |\Psi_i(\bar{z}_j, \bar{x})| \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_\bullet, \delta)} M_i(\bar{x}) \cdot \delta^{m_i + d_i}\right)^{\frac{d_i}{m_i}} \cdot \left(\frac{1}{\epsilon}\right)^{\frac{d_i}{m_i}} < 1 \ .$$

in which case the lower bound on $N_{\epsilon,i}(\bar{z})$ in Theorem 2.2 is either 0 or 1.

2.1.4   Generalizations

The lower bound in Theorem 2.2 can be very weak. In this section we discuss how to calculate a tighter bound.

The construction of Γ in the previous sections holds in any $\bar{B}(\bar{x}_*; \delta)$ satisfying the usual conditions. In particular, if the error function is constructed in each member of a set of nonoverlapping closed balls, then the lower bound on the error is the sum of the errors from each ball.

For fixed i, let $C(\bar{x}; \delta)$ be the open d_i dimensional cube centered on $\bar{x}$ with volume δ^{d_i}. Let $\bar{C}(\bar{x}_*; \delta)$ be its closure. Let $\delta_\nu = 2^{-\nu}$ for some positive integer ν. If $i > 0$, divide I_{d_i} into $\delta_\nu^{-d_i}$ equal sized cubes. Let $X = \{\bar{x}_l\}$ be centers of the cubes. Let $n_{a,l}$ be the number of sampling locations in $C(\bar{x}_l; \delta_\nu)$ associated with an algorithm a. Construct a new error function Γ' in the following way.

a) Let $X' \subset X$ be centers of closed cubes over which $\Psi_i(\bar{z}_j, \bar{x})$ is nonzero and continuous.

b) If $\Psi_i(\bar{z}_j, \bar{x}_l)$ is positive on $C(\bar{x}_l; \delta_\nu)$, then use the previous construction to define a function Γ in the open ball $B(\bar{x}_l; \delta_\nu/2)$. Call this function Γ_l. If $\Psi_i(\bar{z}_j, \bar{x}_l) < 0$, construct the same function Γ, but let $\Gamma_l = -\Gamma$. Note that Γ_l is a member of $G_{a,i,j}(\bar{0})$ in either case.

c) Define Γ' to be

$$\Gamma' = \sum_{\bar{x}_l \in X'} \Gamma_l \ .$$

By the same reasoning used in Lemma 2.8, $\Gamma' \in G_{a,i,j}(\bar{0})$. Thus, a new lower bound on $D_{a,i,j}(\bar{0})/2$ is

$$C_i'' \cdot \delta_\nu^{d_i + m_i} \cdot \sum_{\bar{x}_l \in X'} \left(\min_{\bar{x} \in \bar{B}(\bar{x}_l, \delta_\nu/2)} |\Psi_i(\bar{z}_j, \bar{x})| \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_l, \delta_\nu/2)} M_i(\bar{x}) \cdot \min\{n_{a,l}^{-\frac{m_i}{d_i}}, 1\} \right), \quad (2.6)$$

where $C_i'' = C_i'/2^{m_i + d_i}$. Denote expression 2.6 by $\mathrm{err}_{a,i,\nu,L}(\bar{z}_j)$ To bound $N_{\epsilon,i}(\bar{z}_j)$ for this construction, we need to solve the optimization problem

$$\min_{a \in A_{\epsilon,i}^0(\bar{z}_j)} \sum_{\bar{x}_l \in X'} n_{a,l}$$

subject to

$$\text{err}_{a,i,\nu,L}(\bar{z}_j) \leq \epsilon \quad . \tag{2.7}$$

The optimal sampling locations for the original construction were essentially equidistributed. This construction recognizes that data from different regions of the domain may have differing degrees of influence on the solution, and thus require different amounts of data sampling. This lower bound is a function of the number of subcubes the original cube is divided into, 2^ν. But the solution to problem 2.7 is a lower bound on $N_{\epsilon,i}(\bar{z}_j)$ for any positive integer ν. Thus, the lower bound can be tightened by minimizing the solution for fixed ν over the the set of all possible ν.

2.1.5   Summary of Lower Bound Results

The assumptions on the data functions described in Sections 1.3.3 and 1.3.4 are appropriate for an a priori error analysis of a method whose error is, at least partially, a function of the values of the m_ith order partial derivatives of the data functions. If $\Psi_i(\bar{z}_j, \bar{x})$ and $M_i(\bar{x})$ are both nonzero over an open set in the domain, then the lower bounds derived here imply that the data sampling will increase at least linearly as a function of $\epsilon^{-\frac{d_i}{m_i}}$ when the error tolerance decreases. Note that this bound is on the number of data samples required when approximating a single solution value, and not just when approximating the entire solution function. The construction of the error function described here also allows us to localize the contribution to the error when sampling from any particular subregion.

2.2   Upper Bound

Upper bounds on $N_{\epsilon,i}(\bar{z}_j)$ can be calculated from $N_{a,i}(\bar{z}_j)$ for any particular algorithm $a \in A_\epsilon(\{\bar{z}_j\})$. In particular, for each i, let $\tilde{g}_i$ be some approximation to the data function g_i calculated from values of g_i taken at a finite number sampling locations in I_{d_i}. Define the approximation $\tilde{u}(\bar{z}_j)$ to be the sum of the components $\{\tilde{u}_i(\bar{z}_j)\}$, where each $\tilde{u}_i(\bar{z}_j)$ is defined by

$$\tilde{u}_i(\bar{z}_j) = \int_{I_{d_i}} \Psi_i(\bar{z}_j, \bar{x})\tilde{g}_i(\bar{x})\,d\bar{x} \quad .$$

Then this construction corresponds to an algorithm $a \in A_\epsilon(\{\bar{z}_j\})$ if

$$|u_i(\bar{z}_j) - \tilde{u}_i(\bar{z}_j)| \leq \epsilon/\bar{l}$$

for all i. We will call such an algorithm a *Green's function method* since it corresponds to using the Green's function, and associated kernels, directly. Since

$$
\begin{aligned}
|u_i(\bar{z}_j) - \tilde{u}_i(\bar{z}_j)| &= \left| \int_{I_{d_i}} \Psi_i(\bar{z}_j, \bar{x}) \cdot (g_i(\bar{x}) - \tilde{g}_i(\bar{x})) \, d\bar{x} \right| \\
&\leq \int_{I_{d_i}} |\Psi_i(\bar{z}_j, \bar{x})| \cdot |g_i(\bar{x}) - \tilde{g}_i(\bar{x})| \, d\bar{x} \quad , \qquad (2.8)
\end{aligned}
$$

the solution error can be bounded by a function of the data approximation error.

Most Green's function methods based on piecewise polynomial approximations to the data will satisfy the error criterion if enough equidistributed data sampling locations are used when calculating each component. To verify this, we describe a family of Green's function methods based on a simple piecewise polynomial approximation, and calculate an upper bound on the error for each member of this family. We then use this error bound to calculate an upper bound on $N_{\epsilon,i}(\bar{z}_j)$. Most commonly used algorithms, both Green's function methods and others, will have similar results.

2.2.1 Simple Bounds

Consider a d_i dimensional cube $C(\bar{x}_*; \delta)$. Let $h = \delta/(m_i + 1)$. Let

$$x_{(s,k)} = (\bar{x}_*)_s - \frac{\delta}{2} + kh$$

for $1 \leq s \leq d_i$, where $(\bar{x}_*)_s$ is the sth component of $\bar{x}_*$. Let π_s be the following set of equally spaced locations in $[(\bar{x}_*)_s - \delta/2, (\bar{x}_*)_s + \delta/2]$,

$$\{x_{(s,1)}, \ldots, x_{(s,m_i)}\} \quad .$$

Define $\pi(d_i)$ to be the following set of $m_i^{d_i}$ locations in $C(\bar{x}_*; \delta)$.

$$
\begin{aligned}
\pi(d_i) &= \pi_1 \times \cdots \times \pi_{d_i} \\
&= \{\bar{x} \,|\, (\bar{x})_s \in \pi_s\} \quad . \qquad (2.9)
\end{aligned}
$$

Lemma 2.12 *Let g be a function defined in a d_i dimensional cube $C(\bar{x}_*; \delta)$. Define $\pi(d_i)$ to be the equidistributed locations in $C(\bar{x}_*; \delta)$ described by equation 2.9. Then there exists a unique polynomial $\tilde{g}(\bar{x})$ of degree at most $m_i - 1$ in each variable that interpolates g at the locations in $\pi(d_i)$. If g has m_ith order continous partial derivatives in $C(\bar{x}_*; \delta)$, then the error in the approximation is bounded from above by a term of the form*

$$\tilde{C}_i \cdot \delta^{m_i} \cdot \max_{\bar{x} \in C(\bar{x}_*; \delta)} \max_{\bar{\mu} \in K(m_i)} \left| \frac{\partial^{\bar{\mu}}}{\partial \bar{x}^{\bar{\mu}}} g(\bar{x}) \right| \quad ,$$

where $\tilde{C}_i$ is independent of δ and $\bar{x}_*$.

This is a variation of the results in Chapter 5 of Prenter [Pre75]. See Appendix C for a discussion of the proof.

Let g_i be some element of G_i. Let $C(\bar{x}_*; \delta)$ be some d_i dimensional cube in I_{d_i}. By Lemma 2.12, there exists a unique polynomial that interpolates the function $g_i(\bar{x})$ at the locations $\pi(d_i)$ described above. Call this function $p_i(\bar{x})$. By the equivalence of finite dimensional vector norms [Atk78, page 414], there exists a constant $\tilde{C}_i'$ such that

$$\tilde{C}_i \cdot \max_{\bar{\mu} \in K(m_i)} \left| \frac{\partial^{\bar{\mu}}}{\partial \bar{x}^{\bar{\mu}}} g_i(\bar{x}) \right| \leq \tilde{C}_i' \cdot \| \nabla_{d_i}^{(m_i)} g_i(\bar{x}) \|_{(i)}$$

for all $\bar{x} \in I_{d_i}$, where $\tilde{C}_i$ is defined in Lemma 2.12. Therefore, the error bound on the approximation described in Lemma 2.12 becomes

$$|g_i(\bar{x}) - p_i(\bar{x})| \leq \tilde{C}_i' \cdot \delta^{m_i} \cdot \max_{\bar{x} \in \tilde{C}(\bar{x}_*; \delta)} M_i(\bar{x}) \tag{2.10}$$

when $\bar{x} \in C(\bar{x}_*; \delta)$.

Define an algorithm a in the following way. For each $i > 0$, approximate $u_i(\bar{z}_j)$ in the following way.

a) Divide I_{d_i} into cubes of the form $C(\bar{x}_l; 2^{-\nu_{a,i}})$, for some positive integer $\nu_{a,i}$.

b) In each subcube $C(\bar{x}_l; 2^{-\nu_{a,i}})$, let $p_{a,i,l}(\bar{x})$ be the approximation to $g_i(\bar{x})$ defined in Lemma 2.12. Define $\tilde{g}_{a,i,l}$ to be $p_{a,i,l}$ as a function of $\bar{x} \in I_{d_i}$, $\tilde{g}_{a,i,1}(\bar{x}) = p_{a,i,l}(\bar{x})$.

c) Approximate $u_i(\bar{z}_j)$ by the following expression,

$$\hat{u}_{a,i}(\bar{z}_j) = \sum_l \int_{C(\bar{x}_l; 2^{-\nu_i})} \Psi_i(\bar{z}_j, \bar{x}) \tilde{g}_{i,l}(\bar{x}) \, d\bar{x} \quad .$$

A simple upper bound on the error in this approximation to $u_i(\bar{z}_j)$ follows directly from equations 2.8 and 2.10.

Lemma 2.13

$$|u_i(\bar{z}_j) - \tilde{u}_{a,i}(\bar{z}_j)| \leq \tilde{C}'_i \cdot 2^{-\nu_{a,i} \cdot m_i} \cdot \max_{\bar{x} \in I_{d_i}} M_i(\bar{x}) \cdot \int_{I_{d_i}} |\Psi_i(\bar{z}_j, \bar{x})| \, d\bar{x} \quad .$$

If $i > 0$, then the approximation requires $2^{\nu_{a,i} \cdot d_i}$ subcubes, and a total of $m_i^{d_i} \cdot 2^{\nu_{a,i} \cdot d_i}$ sample locations. The upper bound on $N_{\epsilon,i}(\bar{z}_j)$ generated by algorithms of this type is the following.

Theorem 2.3 *For all* i,

$$N_{\epsilon,i}(\bar{z}_j) \leq \tilde{C}''_i(\bar{z}_j) \cdot \max_{\bar{x} \in I_{d_i}} M_i^{\frac{d_i}{m_i}}(\bar{x}) \cdot \left(\frac{1}{\epsilon}\right)^{\frac{d_i}{m_i}} + 1 \quad ,$$

for a finite constant $\tilde{C}''_i(\bar{z}_j)$ *independent of* M_i *and* ϵ.

Proof: Consider an algorithm a satisfying the conditions on page 50, where cubes of the form $C(\bar{x}_l; 2^{-\nu_{a,i}})$ are used to subdivide I_{d_i} when approximating the ith component. Therefore,

$$N_{a,i}(\bar{z}_j) = m_i^{d_i} \cdot 2^{(\nu_{a,i}) \cdot d_i} \quad . \tag{2.11}$$

From Lemma 2.13, a sufficient condition to satisfy the error criterion is

$$\tilde{C}'_i \cdot 2^{-\nu_{a,i} \cdot m_i} \cdot \max_{\bar{x} \in I_{d_i}} M_i(\bar{x}) \int_{I_{d_i}} |\Psi_i(\bar{z}_j, \bar{x})| \, d\bar{x} \leq \frac{\epsilon}{\bar{l}}$$

for all i. Therefore,

$$\left(\frac{\bar{l} \cdot \tilde{C}'_i \cdot \max_{\bar{x} \in I_{d_i}} M_i(\bar{x}) \int_{I_{d_i}} |\Psi_i(\bar{z}_j, \bar{x})| \, d\bar{x}}{\epsilon} \right)^{\frac{d_i}{m_i}} \leq 2^{\nu_{a,i} \cdot d_i} \tag{2.12}$$

for all i is also a sufficient condition. Define a so that

$$\nu_{a,i} = \left\lfloor \frac{1}{m_i} \cdot \log_2 \left(\frac{\bar{l} \cdot \tilde{C}'_i \cdot \max_{\bar{x} \in I_{d_i}} M_i(\bar{x}) \int_{I_{d_i}} |\Psi_i(\bar{z}_j, \bar{x})| \, d\bar{x}}{\epsilon} \right) \right\rfloor + 1 \tag{2.13}$$

for all i. Then the bounds in inequality 2.12 are satisfied, and $a \in A_\epsilon(\{\bar{z}_j\})$. Since $N_{a,i}(\bar{z}_j) \geq N_{\epsilon,i}(\bar{z}_j)$ when $a \in A_\epsilon(\{\bar{z}_j\})$, the Theorem holds for

$$\tilde{C}_i''(\bar{z}_j) = 2^{d_i} \cdot m_i^{d_i} \cdot \left(\bar{l} \cdot \tilde{C}_i' \cdot \int_{I_{d_i}} |\Psi_i(\bar{z}_j, \bar{x})| \, d\bar{x} \right)^{\frac{d_i}{m_i}} \ .$$

This follows directly from equations 2.11 and 2.13. Since $\Psi_i(\bar{z}_j, \bar{x}) \in L^1(I_{d_i})$, this constant is finite. ∎

As defined in Section 1.3.7, $f(x) = \Theta(g(x))$ if there exist positive constants c_1, c_2, and x_0 such that

$$c_1 |g(x)| \leq |f(x)| \leq c_2 |g(x)|$$

for all $x > x_0$ [HS78]. Using this notation, Theorems 2.2 and 2.3 imply the following corollary.

Corollary 2.4 *If there exists a closed ball $\bar{B}(\bar{x}_*; \delta) \subset I_{d_i}$ on which $M_i(\bar{x})$ is nonzero, and on which $\Psi_i(\bar{z}_j, \bar{x})$ is nonzero and continuous, then*

$$N_{\epsilon,i}(\bar{z}_j) = \Theta \left(\epsilon^{-\frac{d_i}{m_i}} \right)$$

as a function of $1/\epsilon$.

Proof: If $d_i = 0$, then Theorem 2.2 states that $N_{\epsilon,i}(\bar{z}_j) = 1$ for these assumptions, and the statement holds. If $d_i > 0$, then Theorem 2.2 bounds $N_{\epsilon,i}(\bar{z}_j)$ from below by

$$N_{\epsilon,i}(\bar{z}_j) \geq c_1(\bar{z}_j) \cdot \left(\frac{1}{\epsilon} \right)^{\frac{d_i}{m_i}} \ ,$$

where

$$c_1(\bar{z}_j) = \left[C_i' \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_*, \delta)} |\Psi_i(\bar{z}_j, \bar{x})| \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_*, \delta)} M_i(\bar{x}) \cdot \delta^{m_i + d_i} \right]^{\frac{d_i}{m_i}} \ .$$

Theorem 2.3 bounds $N_{\epsilon,i}(\bar{z}_j)$ from above by

$$N_{\epsilon,i}(\bar{z}_j) \leq c_2(\bar{z}_j) \cdot \left(\frac{1}{\epsilon} \right)^{\frac{d_i}{m_i}} + 1 \ ,$$

where

$$c_2(\bar{z}_j) = \tilde{C}_i''(\bar{z}_j) \cdot \max_{\bar{x} \in I_{d_i}} M_i^{\frac{d_i}{m_i}}(\bar{x}) \ .$$

Let $c_2'(\bar{z}_j) = c_2(\bar{z}_j) + 1$. Then

$$c_1(\bar{z}_j) \cdot \left(\frac{1}{\epsilon}\right)^{\frac{d_i}{m_i}} \leq N_{\epsilon,i}(\bar{z}_j) \leq c_2'(\bar{z}_j) \cdot \left(\frac{1}{\epsilon}\right)^{\frac{d_i}{m_i}}$$

when $1/\epsilon > 1$. This proves the corollary. ∎

2.2.2 Improved Bounds

The bounds in the previous lemma and theorem can be improved by allowing the sampling frequency to vary over the domain. For example, define a Green's function method a in the following way. Divide the domain I_{d_i} into cubes of the form $C(\bar{x}_l; 2^{-\nu})$ as in Section 2.1.4. Let X be the set of centers of these cubes. Divide each of these cubes into cubes of the form $C(\bar{x}_{l'}; 2^{-\nu_{a,l}})$ where $\nu_{a,l} \geq \nu$. Use the polynomial approximation from Section 2.2.1 in each of these smaller cubes to define an approximation to g_i in I_{d_i}. Call this approximation $\tilde{g}_i$. By Lemma 2.12 and inequality 2.10, an upper bound on the error in using $\int_{I_{d_i}} \Psi_i(\bar{z}_j, \bar{x}) \tilde{g}_i(\bar{x}) \, d\bar{x}$ to approximate $u_i(\bar{z}_j)$ is

$$\tilde{C}_i' \cdot \sum_{\bar{x}_l \in X} \left(2^{-\nu_l \cdot m_i} \cdot \max_{\bar{x} \in \tilde{C}(\bar{x}_l; 2^{-\nu}) \cap I_{d_i}} M_i(\bar{x}) \cdot \int_{C(\bar{x}_l; 2^{-\nu}) \cap I_{d_i}} |\Psi_i(\bar{z}_j, \bar{x})| \, d\bar{x} \right) .$$

$$(2.14)$$

Denote this expression by $\text{err}_{a,i,\nu,U}(\bar{z}_j)$.

For this construction, the number of sample locations in $C(\bar{x}_l; 2^{-\nu})$ is

$$m_i^{d_i} \cdot 2^{(\nu_l - \nu) \cdot d_i} .$$

The total number of sample locations is

$$m_i^{d_i} \cdot \sum_{\bar{x}_l \in X} 2^{(\nu_l - \nu) \cdot d_i} .$$

To minimize the total number of sampling locations in I_{d_i} while still preserving the error bound, solve the problem

$$\min \; m_i^{d_i} \cdot \sum_{\bar{x}_l \in X} 2^{(\nu_l - \nu) \cdot d_i}$$

subject to

$$\text{err}_{a,i,\nu,U}(\bar{z}_j) \leq \frac{\epsilon}{\bar{l}} \ .$$

The minimization is taken over all legal combinations of the $\{\nu_l\}$ values. The solution to this problem indicates how many subcubes to place in each of the larger cubes. This is equivalent to specifying the number of sample locations to place in the larger cubes.

Using equation 2.14, we can close a gap left in the lower bound analysis. We have no bounds on $N_{\epsilon,i}(\bar{z}_j)$ for the case when $M_i(\bar{x}) \cdot \Psi_i(\bar{z}_j, \bar{x}) = 0$ for all $\bar{x} \in I_{d_i}$, but neither function is identically zero. If $\nu_l = \nu$ for all l in equation 2.14, then an immediate consequence of this construction is the following theorem.

Theorem 2.5 *For any i, assume that either $M_i(\bar{x}) \equiv 0$ or $\Psi_i(\bar{z}_j, \bar{x}) \equiv 0$ in each subregion $C(\bar{x}_l; 2^{-\nu}) \cap I_{d_i}$, $\bar{x}_l \in X$. Then*

$$N_{\epsilon,i}(\bar{z}_j) \leq m_i^{d_i} \cdot 2^{\nu \cdot d_i} \ .$$

Proof: Pick some $\epsilon > 0$. Let i' be some element of $\{0, \ldots, \bar{l}\}$. Define a to be an algorithm that approximates $u(\bar{z}_j)$ in the following way. Approximate $u_{i'}(\bar{z}_j)$ in the fashion described above, using $\nu_l = \nu$. Approximate the other components in the fashion described on page 50, where $\nu_{a,i}$ satisfies equation 2.13 for all $i \neq i'$. Thus, $|u_i(\bar{z}_j) - \tilde{u}_{a,i}(\bar{z}_j)| \leq \epsilon/\bar{l}$ when $i \neq i'$. Assume that, for the given subdivision of I_{d_i}, either $M_i(\bar{x}) \equiv 0$ or $\Psi_i(\bar{z}_j, \bar{x}) \equiv 0$ in each subregion $C(\bar{x}_l; 2^{-\nu}) \cap I_{d_i}$, $\bar{x}_l \in X$. Then

$$|u_{i'}(\bar{z}_j) - \tilde{u}_{a,i'}(\bar{z}_j)| \leq \text{err}_{a,i',\nu,U}(\bar{z}_j) = 0 \ .$$

Therefore $a \in A_\epsilon(\{\bar{z}_j\})$ and

$$N_{a,i'}(\bar{z}_j) = m_i^{d_i} \cdot 2^{\nu \cdot d_i} \ .$$

Since $N_{a,i'}(\bar{z}_j)$ is an upper bound on $N_{\epsilon,i'}(\bar{z}_j)$, and since both ϵ and i' were arbitrary, this proves the Theorem. ∎

Therefore, $N_{\epsilon,i}(\bar{z}_j)$ is a bounded function of ϵ if the assumption in Theorem 2.5 holds for any finite ν.

2.2.3 Summary of Upper Bound Results

Upper bounds on the amount of data needed when using simple piecewise polynomial approximations to a data function have the same asymptotic behavior as the lower bound calculated in Section 2.1. Therefore, only constants can be improved by using global approximation methods to approximate these data functions. For polynomial approximations with local bases [SF73], like the example provided here, the contribution to the error from a subregion is a function of the number of sample locations in that subregion. This allows us to decrease the upper bound by adjusting the frequency of sampling in subregions.

2.3 Optimal Parallel Algorithms

In this section we calculate lower bounds on the complexity of parallel implementations of algorithms in $A_\epsilon(Z)$. Assume that we have a fully connected multiprocessor with an unlimited number of homogeneous processors and memories. Furthermore, assume that it has infinitely powerful communication capabilites. That is, the memory access and transmission times are both zero. Thus, the communication cost is always zero on this architecture. Consider the class of all parallel implementations of algorithms in $A_\epsilon(Z)$ on this idealized multiprocessor. Then the minimum parallel complexity achieved by this class represents a lower bound on the parallel cost for all multiprocessors in the following sense. Assume that the processors in a given multiprocessor are no faster than those in the idealized multirpocessor. Then any parallel algorithm on the given multiprocessor can be simulated on the idealized multiprocessor with no increase in the parallel complexity. If some processors are faster, then the bound is recovered by scaling the complexity to reflect this difference.

We will refer to the set of parallel algorithms that achieve this minimum as *optimal parallel algorithms*. Note that the serial algorithms corresponding to optimal parallel algorithms may not have good parallel implementations on a particular multiprocessor. The finite number of processors and the cost of communication can change the behavior of the parallel cost. In this section we describe what form this lower bound takes, and describe a class of simple linear algorithms whose parallel complexity has the same asymptotic

behavior as $\epsilon \to 0$.

2.3.1 Lower Bound on Parallel Complexity

Lemma 2.1 provides a lower bound on the parallel complexity of an optimal parallel algorithm. The following theorem then follows from Theorem 2.2.

Theorem 2.6 *Assume that there exists some $\bar{z} \in Z$ and some $i \in \{1,\ldots,\bar{l}\}$ such that $d_i \neq 0$ and the following condition holds. There exists a closed ball $\bar{B}(\bar{x}_*; \delta) \subset I_{d_i}$ on which $\Psi_i(\bar{z}, \bar{x})$ is nonzero and continuous, and on which $M_i(\bar{x})$ is nonzero. Then the parallel complexity of an optimal parallel algorithm for $A_\epsilon(Z)$ is bounded from below by an expression of the form*

$$f_{(+)} \cdot \left(c_1 \ + \ c_2 \cdot \log_2 \left(\frac{1}{\epsilon} \right) \right)$$

as $\epsilon \to 0$. The constants c_1 and c_2 are both independent of ϵ, and $c_2 > 0$.

Proof: Let

$$\epsilon_* = C_i \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_*, \delta)} |\Psi_i(\bar{z}_j, \bar{x})| \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_*, \delta)} M_i(\bar{x}) \cdot \delta^{m_i + d_i} \ .$$

By Theorem 2.2, if $\epsilon < \epsilon_*$, then

$$N_{a,i}(\bar{z}) \ \geq \ \left[C_i' \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_*, \delta)} |\Psi_i(\bar{z}, \bar{x})| \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_*, \delta)} M_i(\bar{x}) \cdot \delta^{m_i + d_i} \right]^{\frac{d_i}{m_i}} \cdot \left(\frac{1}{\epsilon} \right)^{\frac{d_i}{m_i}}$$

for all $a \in A_\epsilon(Z)$. Therefore, if $\epsilon < \epsilon_*$, then

$$\max_{\bar{z} \in Z} \lceil \log_2 (N_{a,i}(\bar{z})) \rceil \ \geq \ c_1 + \frac{d_i}{m_i} \cdot \log_2 \left(\frac{1}{\epsilon} \right) \ ,$$

where

$$c_1 = \max_{\bar{z} \in Z} \frac{d_i}{m_i} \cdot \log_2 \left(C_i' \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_*, \delta)} |\Psi_i(\bar{z}, \bar{x})| \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_*, \delta)} M_i(\bar{x}) \cdot \delta^{m_i + d_i} \right) \ .$$

By the assumptions of the Theorem, c_1 is finite. The proof then follows from Lemma 2.1.
∎

Corollary 2.7 *Given the assumptions of Theorem 2.6, the parallel complexity of parallel implementations of algorithms in $A_\epsilon(Z)$ grow at least linearly as a function of $\log_2(\epsilon^{-1})$ when ϵ decreases.*

2.3.2 Upper Bound on Parallel Complexity

In Theorem 2.3 we described an algorithm $a \in A_\epsilon(\{\bar{z}\})$ that approximated each component of $u(\bar{z})$ by an expression of the form

$$
\begin{aligned}
\tilde{u}_{a,i}(\bar{z}) &= \int_{I_{d_i}} \Psi_i(\bar{z},\bar{x}) \tilde{g}_{a,i}(\bar{x}) \, d\bar{x} \\
&= \sum_{l=1}^{2^{\nu_{a,i} \cdot d_i}} \int_{C(\bar{x}_l; 2^{-\nu_{a,i} \cdot d_i})} \Psi_i(\bar{z},\bar{x}) \tilde{g}_{a,i,l}(\bar{x}) \, d\bar{x}
\end{aligned}
\tag{2.15}
$$

when $i > 0$. Each function $\tilde{g}_{a,i,l}(\bar{x})$ is the polynomial interpolant of $g(\bar{x})$ in the d_i dimensional cube $C(\bar{x}_l; 2^{-\nu_{a,i} \cdot d_i})$ described in Section 2.2.1.

Let $\pi_l(d_i) = \{\bar{x}_{l,k}\}$ be the set of sampling locations in $C(\bar{x}_l; 2^{-\nu_{a,i} \cdot d_i})$. Let $l_{l,k}(\bar{x})$ be the unique polynomial of degree at most $m_i - 1$ in each variable such that

$$
l_{l,k}(\bar{x}) = \begin{cases} 1, & \text{if } \bar{x} = \bar{x}_{l,k} ; \\ 0, & \text{if } \bar{x} \in \pi_l(d_i), \ \bar{x} \neq \bar{x}_{l,k} . \end{cases}
$$

Then $\tilde{g}_{a,i,l}(\bar{x})$ can be represented by

$$
\sum_{k=1}^{m_i^{d_i}} g_i(\bar{x}_{l,k}) \cdot l_{l,k}(\bar{x}) .
$$

See AppendixC and [Pre75] for a more detailed discussion of this representation of the interpolating polynomial.

The expression for $\tilde{u}_{a,i}(\bar{x})$ in equation 2.15 becomes

$$
\begin{aligned}
\tilde{u}_{a,i}(\bar{z}) &= \sum_{l=1}^{2^{\nu_{a,i} \cdot d_i}} \int_{C(\bar{x}_l; 2^{-\nu_{a,i} \cdot d_i})} \Psi_i(\bar{z},\bar{x}) \left(\sum_{k=1}^{m_i^{d_i}} g_i(\bar{x}_{l,k}) \cdot l_{l,k}(\bar{x}) \right) d\bar{x} \\
&= \sum_{l=1}^{2^{\nu_{a,i} \cdot d_i}} \sum_{k=1}^{m_i^{d_i}} g_i(\bar{x}_{l,k}) \cdot \int_{C(\bar{x}_l; 2^{-\nu_{a,i} \cdot d_i})} \Psi_i(\bar{z},\bar{x}) l_{l,k}(\bar{x}) \, d\bar{x} .
\end{aligned}
$$

Each factor $\int_{C(\bar{x}_l; 2^{-\nu_{a,i} \cdot d_i})} \Psi_i(\bar{z},\bar{x}) l_{l,k}(\bar{x}) \, d\bar{x}$ is independent of the data, and can be precomputed. Call it $r_{a,i,l,k}(\bar{z})$. The expression for $\tilde{u}_{a,i}(\bar{z})$ is then

$$
\tilde{u}_{a,i}(\bar{z}) = \sum_{l=1}^{2^{\nu_{a,i} \cdot d_i}} \sum_{k=1}^{m_i^{d_i}} r_{a,i,l,k}(\bar{z}) \cdot g_i(\bar{x}_{l,k}) .
\tag{2.16}
$$

The multiplications in expression 2.16 are all independent, and the additions can be calculated in parallel as in the example on page 10. Therefore, the parallel complexity of evaluating this expression can be as low as

$$f_{(*)} + f_{(+)} \cdot \left\lceil \log_2 \left(m_i^{d_i} \cdot 2^{\nu_{a,i} \cdot d_i} \right) \right\rceil \tag{2.17}$$

when $m_i^{d_i} \cdot 2^{\nu \cdot d_i}$ processors are used. Let

$$\hat{C}_i(\bar{z}) = \tilde{C}_i''(\bar{z}) \cdot \max_{\bar{x} \in I_{d_i}} M_i^{\frac{d_i}{m_i}}(\bar{x}) \cdot$$

$$= 2^{d_i} \cdot m_i^{d_i} \cdot \max_{\bar{x} \in I_{d_i}} M_i^{\frac{d_i}{m_i}}(\bar{x}) \cdot \left(\bar{l} \cdot \tilde{C}_i' \cdot \int_{I_{d_i}} |\Psi_i(\bar{z}_j, \bar{x})| \, d\bar{x} \right)^{\frac{d_i}{m_i}} , \tag{2.18}$$

where $\tilde{C}_i''(\bar{z}_j)$ is the constant defined in Theorem 2.3 and $\tilde{C}_i'$ is the constant from equation 2.10 and Lemma 2.13. By the definition of a in Theorem 2.3, we know that

$$\log_2 \left(m_i^{d_i} \cdot 2^{\nu_{a,i} \cdot d_i} \right) \leq \log_2 \left(\hat{C}_i(\bar{z}) \cdot \left(\frac{1}{\epsilon} \right)^{\frac{d_i}{m_i}} + 1 \right)$$

$$\leq \max \left\{ \log_2 \hat{C}_i(\bar{z}) + \frac{d_i}{m_i} \cdot \log_2 \left(\frac{1}{\epsilon} \right) + 1, 0 \right\} . \tag{2.19}$$

Since Ψ_i is a type 1 kernel, $\hat{C}_i(\bar{z})$ is a finite nonnegative constant independent of ϵ.

The approximation of $u(\bar{z})$ requires the evaluation of an expression like expression 2.16 for each component. The final step of the approximation is

$$\tilde{u}_a(\bar{z}) = \sum_{i=1}^{\bar{l}} \tilde{u}_{a,i}(\bar{z}) .$$

The parallel complexity of evaluating this sum can be as low as

$$f_{(+)} \cdot \left\lceil \log_2 \bar{l} \right\rceil \tag{2.20}$$

if $\lfloor \bar{l}/2 \rfloor$ processors are used. The following lemma describes how small the parallel complexity of algorithm a can be on the idealized multiprocessor.

Lemma 2.14 *Approximate $u(z)$ to within an error tolerance of ϵ in the following way. Use the algorithm on page 50 of Section 2.2.1 to approximate each component*

$u_i(\bar{z})$ *to within an error tolerance of $\epsilon/\bar{l}$ by satisfying equation 2.13. There exists a parallel implementation of this algorithm whose parallel complexity is bounded from above by an expression of the form*

$$f_{(\cdot)} + f_{(+)} \cdot \left(c_3(\bar{z}) \; + \; c_4 \cdot \log_2 \left(\frac{1}{\epsilon} \right) \right)$$

when $\epsilon < 1$. The constants $c_3(\bar{z})$ and c_4 are independent of ϵ, and c_4 is nonnegative. No more than

$$(\bar{l} + 1) \; + \; \sum_{i=1}^{\bar{l}} \hat{C}_i(\bar{z}) \cdot \left(\frac{1}{\epsilon} \right)^{\frac{d_i}{m_i}}$$

processors are required to achieve this bound. Each $\hat{C}_i(\bar{z})$ is a finite nonnegative constant independent of ϵ.

Proof: Since the approximation of each component is independent of the others, the parallel complexity of approximating the components can be as low as

$$f_{(\cdot)} \; + \; \max_{i \in \{1,\dots,\bar{l}\}} \left(\left\lceil \max \left\{ \log_2 \hat{C}_i(\bar{z}) + \frac{d_i}{m_i} \cdot \log_2 \left(\frac{1}{\epsilon} \right) + 1 \right\rceil , \, 0 \right\} \right)$$

if $\lfloor \hat{C}_i(\bar{z}) \cdot (\epsilon)^{-\frac{d_i}{m_i}} + 1 \rfloor$ processors are used to calculate each component. Therefore,

$$\bar{l} \; + \; \sum_{i=1}^{\bar{l}} \hat{C}_i(\bar{z}) \cdot \left(\frac{1}{\epsilon} \right)^{\frac{d_i}{m_i}}$$

processors are enough for this step of the parallel implementation. This follows from equations 2.17 and 2.19. Using expression 2.20, the final summation of the components need only have a parallel complexity of $f_{(+)} \cdot \lceil \log_2 \bar{l} \rceil$ when $\lfloor \bar{l}/2 \rfloor$ processors are used. Enough processors are available from the previous step to satisfy this condition. Thus, the Lemma holds if

$$c_3(\bar{z}) = \left\lceil \log_2 \bar{l} \right\rceil \; + \; \max_{i \in \{1,\dots,\bar{l}\}} \left(\max \left\{ \log_2 \hat{C}_i(\bar{z}) + 1 , \, 0 \right\} \right) \; + \; 1 \; ,$$

and

$$c_4 \; = \; \max_{i \in \{1,\dots,\bar{l}\}} \left(\frac{d_i}{m_i} \right) \quad .$$

∎

The following theorem is an immediate consequence of this lemma.

Theorem 2.8 *Define an algorithm $a \in A_\epsilon(Z)$ in the following way. For each $\bar{z} \in Z$, approximate $u(\bar{z})$ to within an error tolerance of ϵ using the algorithm in Lemma 2.14. There exists a parallel implementation of a whose parallel complexity is bounded from above by an expression of the form*

$$f_{(*)} + f_{(+)} \cdot \left(c_3 + c_4 \cdot \log_2\left(\frac{1}{\epsilon}\right)\right) \quad .$$

The constants c_3 and c_4 are both independent of ϵ, and $c_4 > 0$. Let $|Z|$ be the number of locations in Z. There exists an expression

$$|Z| \cdot \left(\sum_{i=1}^{\bar{l}} \hat{C}_i \cdot \left(\frac{1}{\epsilon}\right)^{\frac{d_i}{m_i}} + \bar{l}\right)$$

that is an upper bound on the number of processors required to achieve this bound on the complexity, where each $\hat{C}_i$ is a finite nonnegative constant independent of ϵ.

Proof: The approximation of each solution value $u(\bar{z})$ for $\bar{z} \in Z$ is independent of the others. Thus, we can calculate each solution value on a different subset of the processors in the idealized multiprocessor. The bound follows from Lemma 2.14 when

$$c_3 = \max_{\bar{z} \in Z} c_3(\bar{z}) \quad ,$$

$$\hat{C}_i = \max_{\bar{z} \in Z} \hat{C}_i(\bar{z}) \text{ for } i \in \{1, \ldots, \bar{l}\} \quad ,$$

and c_4 is the same as in Lemma 2.14. By assumption 2 on page 18,

$$\max_{\bar{z} \in Z} \int_{I_{d_i}} |\Psi_i(\bar{z}, \bar{x})| \, d\bar{x} \leq \max_{\bar{z} \in \Omega} \int_{I_{d_i}} |\Psi_i(\bar{z}_j, \bar{x})| \, d\bar{x} \leq \infty \quad .$$

That $\hat{C}_i$ is nonnegative and finite then follows from equation 2.18 and the assumptions on $M_i(\bar{x})$. ∎

Corollary 2.9 *The parallel complexity of an optimal parallel algorithm for $A_\epsilon(Z)$ is bounded from above by an expression of the form*

$$f_{(*)} + f_{(+)} \cdot \left(c_3 + c_4 \cdot \log_2\left(\frac{1}{\epsilon}\right)\right) \quad .$$

The constants c_3 and c_4 are both independent of ϵ, and $c_4 > 0$.

Proof: The algorithm described in Theorem 2.8 is a member of $A_\epsilon(Z)$. Therefore, the upper bound described in Theorem 2.8 is also an upper bound on the parallel complexity of the optimal parallel algorithms. ∎

Note that using the algorithm in Section 2.2.1 for each component of each solution value in $\{u(\bar{z}) \,|\, \bar{z} \in Z\}$ describes a finite complexity[1] explicit linear algorithm of the type defined in Section 1.3.7,

$$\tilde{u} = \sum_{i=1}^{I} R_i \tilde{g}_i \quad . \tag{2.21}$$

Assume that the jth element of the vector $\tilde{u}$ approximates $u(\bar{z}_j)$ for $\bar{z}_j \in Z$. Then, for fixed i, the elements of the set $\{r_{i,l,k}(\bar{z}_j)\}$ are the nonzero elements in the jth row of R_i. The following corollary is an immediate consequence of this discussion and Theorem 2.9.

Corollary 2.10 *There exists a finite constant C independent of ϵ for which the following property holds: For any $\epsilon > 0$, there exists a finite complexity explicit linear algorithm $a \in A_\epsilon(Z)$ with a parallel implementation whose parallel complexity is bounded from above by an expression of the form*

$$f_{(*)} \; + \; C \cdot f_{(+)} \cdot \log_2 \frac{1}{\epsilon} \quad .$$

Only a finite number of processors are required to achieve this bound.

2.3.3 Summary of Complexity Bounds

When the conditions assumed in Theorem 2.6 are satisfied, the parallel complexity of all parallel algorithms must increase as the error tolerance decreases. And, as $\epsilon \to 0$, there is always a parallel implementation of an explicit linear algorithm in $A_\epsilon(Z)$ whose parallel complexity is within a constant factor of the lower bound. The size of this constant factor can be quite large. But the size of the factor can be reduced by using the generalizations described in Sections 2.1.4 and 2.2.2.

---

[1]The finite complexity of this algorithm is easily verified directly. It is also a consequence of Theorem 2.8. The serial complexity is bounded from above by the product of the parallel complexity and the number of processors used. Both of these are finite.

2.4   Conclusion

In this chapter we have described upper and lower bounds on $N_{\epsilon,i}(\bar{z}_j)$.

- If $\Psi_i(\bar{z}_j, \bar{x})$ is not nonzero over some open set contained in I_{d_i}, then $N_{\epsilon,i}(\bar{z}_j) = 0$.

- If $N_{\epsilon,i}(\bar{z}_j) \neq 0$ and $d_i = 0$, then $N_{\epsilon,i}(\bar{z}_j) = 1$.

- If there exists a finite covering of the domain by d_i dimensional cubes such that either $\Psi_i(\bar{z}_j, \bar{x}) \equiv 0$ or $M_i(\bar{x}) \equiv 0$ in each cube, then $N_{\epsilon,i}(\bar{z}_j)$ is bounded independent of ϵ.

- If $\Psi_i(\bar{z}_j, \bar{x})$ and $M_i(\bar{x})$ are both nonzero on any open set in the domain, then

$$N_{\epsilon,i}(\bar{z}_j) = \Theta\left(\epsilon^{-\frac{d_i}{m_i}}\right)$$

  as a function of $1/\epsilon$.

We have also described upper and lower bounds on the asymptotic behavior of the parallel complexity as the error tolerance decreases.

- If there exists at least one $\bar{z} \in Z$ and one $i \in \{1, \ldots, \bar{l}\}$ such that $\Psi_i(\bar{z}_j, \bar{x})$ and $M_i(\bar{x})$ are both nonzero on any open set in the domain, then the parallel complexity of parallel implementations of algorithms in $A_\epsilon(Z)$ must increase at least linearly as a function of

$$f_{(+)} \cdot d_i \cdot \log_2\left(\frac{1}{\epsilon}\right)$$

  when $\epsilon \to 0$.

- For an ideal multiprocessor with arbitrarily many processors and infinitely powerful communication capabilities, the parallel complexity of an optimal parallel algorithm is bounded above by a function of the form

$$f_{(+)} \cdot \max_{i \in \{1, \ldots, \bar{l}\}} \left(\frac{d_i}{m_i} \log_2\left(\frac{1}{\epsilon}\right)\right) + c \ ,$$

  where c is a finite constant independent of ϵ.

Chapter 3

Scaling

Current research into parallel processing and multiprocessors is driven by the need to increase computing power. Two goals are achieved by increasing the computing power. More problems can be solved in a given interval of time and problems whose solutions have heretofore been too costly to calculate can now be solved.

In this chapter we examine an issue related to how effective multiprocessors are at achieving these goals, *scaling*. A multiprocessor scales if increasing the number of processors enables it to solve larger problems efficiently. A lack of parallelism in the algorithms or increasing communication cost can prevent this. We analyze this issue by using the results of Chapter 2 to calculate lower bounds on the parallel cost of approximating the solution of a PDE on a particular multiprocessor. We bound the parallel cost as the serial complexity of calculating this approximation increases. We also examine the effect of communication costs on this bound.

3.1   Definition of Scaling

Scaling a multiprocessor architecture increases or decreases the number of processors and memories in the multiprocessor while keeping certain attributes of the architecture fixed.[1] In particular, we define scaling for the example architectures of Section 1.1.3 in the

---

[1]See [LM87] for a discussion of parameterized architectures. These are families of architectures whose individual members are specified by a particular choice of the parameters, like the number of

following way. The graph of the scaled multiprocessor architecture is of the same type as before scaling, and each type of component has the same parameters. For example, when scaling a d dimensional array multiprocessor with N^d processors, the graph of the new multiprocessor will still be a d dimensional cube. The graph labels indicating processor and communication abilities will also be unchanged. But the number of processors will now be M^d for some $M \neq N$. Thus, scaling defines a family of multiprocessors with similar architectures. We will refer to a particular multiprocessor as an *instance* of this architectural family and to the number of processors in an instance as its *size*. We will use the term *scaling up* to mean increasing the size of a multiprocessor architecture.

Scaling a problem alters the problem specifications in such a way that the serial complexity of the algorithms used to solve it changes. By the assumptions in Section 1.3.6, the problem of approximating the PDE is a function of the number and location of solution values to be approximated, the available data, and the bound on the error in approximating the solution values. Therefore, to scale the problem, one or more of these parameters must be changed. Scaling the problem defines a family of similar problems, all approximating the solution of the PDE. We will refer to a given set of specifications as a problem *instance*. The *size* of a problem instance is the minimum serial complexity of algorithms that solve the problem instance. We will use the term *scaling up* to mean changing the problem specifications in such a way that its size increases.

We say that a multiprocessor architecture *scales* for a problem if the minimum parallel cost can be bounded independent of the problem size by scaling up the architecture. That is, there exists a finite constant associated with the problem and the architecture with the following property. For any given problem instance, there is always some instance of the architecture that is large enough that the minimum parallel cost is less than this constant. Given a reasonable assumption on how the size of a problem can grow, we will show that no multiprocessor architecture scales for our problems.

---

processors. In general, we can define a scaling of a given multiprocessor by replicating the original multiprocessor some number of times, and joining these pieces together in some consistent way.

3.2 Problem Scaling Assumptions

We will not increase the size of the problem unless some advantage is gained by doing so. On serial computers, there are only two possible reasons for increasing the size of problem. First, we may be interested in approximating the solution more accurately, and are willing to pay the price of a longer execution time. Second, the algorithm we are using may not be optimal, and solving a problem instance with a larger size may take less time than when solving the original instance. But, as the size continues increasing, the increasing minimum serial complexity will force the serial complexity of all algorithms to eventually increase. Thus, ignoring the vagaries of individual algorithms, the most common reason for increasing the size of the problem is to improve the approximation to the solution.

On multiprocessors, there is one additional reason to increase the size of the problem. It may be that a problem instance with a larger size can be solved with a smaller parallel cost than the original probem instance could be. This may be due to the vagaries of a given parallel algorithm, or to a smaller minimum parallel cost. But, for a fixed error bound ϵ, there is a lower bound on the minimum parallel cost that is independent of the other factors, as was described in chapters 1 and 2. Therefore, as the problem continues to scale up, any decrease in the parallel cost will be limited by the lower bound, and any advantage to scaling up the problem will suffer from a law of diminishing returns. Additionally, a decrease in the minimum parallel cost is unlikely when scaling up the problem, and would be difficult to predict. In practice, when a problem is scaled up, it is done in order to improve the approximation to the solution.

Given the above discussion, we will assume that we are only interested in scaling up a problem if the approximation to the solution is thereby improved. Our first assumption specifies how we will measure this improvement. This assumption includes the most commonly used techniques.

i) *If the problem is scaled up, then some upper bound on the error in approximating the solution of the PDE is decreased. We will assume two different types of error bounds.*

   a) *Let Z be a finite set of locations in Ω, and assume that the error bound to*

be satisfied is of the form

$$\max_{u \in U} \max_{\bar{z}_j \in Z} |u(\bar{z}_j) - \tilde{u}(\bar{z}_j)| \leq \epsilon .$$

For each $\bar{z}_j \in Z$ and $u \in U$, $\tilde{u}(\bar{z}_j)$ is an approximation to $u(\bar{z}_j)$ calculated from the data specified by the problem instance.

b) *Assume that the error bound is on how well the solution function is approximated,*

$$\max_{u \in U} \max_{\bar{z} \in \Omega} |u(\bar{z}) - \tilde{u}(\bar{z})| \leq \epsilon .$$

For each $u \in U$, the function $\tilde{u}$ is an approximation to u satisfying the following property. For each problem instance, $\tilde{u}(\bar{z})$ is a function of a finite set of values $\{\tilde{u}(\bar{z}_j)\}$, where $\{\bar{z}_j\}$ is the set of locations specified by the instance and each $\tilde{u}(\bar{z}_j)$ is an approximation to $u(\bar{z}_j)$ calculated from the available data. Furthermore, for each $\bar{z} \in \Omega$, $\tilde{u}(\bar{z})$ is a function of at most c of these approximate solution values, where c is independent of the size of the problem and $\bar{z}$.

For case b, $\tilde{u}$ might be a piecewise polynomial representation of the function based on the set of values specified by the problem instance. The assumption about how $\tilde{u}(\bar{z})$ is defined is to ensure that approximating the solution at the locations specified by the problem instance represents the major component of the cost of approximating the function.

We also want the increase in the size to be effective in decreasing the error. This motivates the second assumption about how the problem scales.

2) *There exists a monotonically increasing continuous function η with the following properties.*

   a) *$\eta(x)$ is finite and positive for finite $x > 0$, and $\eta(0) = 0$.*

   b) *Let $s(\mathbf{p})$ be the size of a problem instance $\mathbf{p}$. Let $\epsilon_{\mathbf{p}}$ be the error bound satisfied by the instance. Then,*

$$\epsilon_{\mathbf{p}} \leq \eta\left(\frac{1}{s(\mathbf{p})}\right) .$$

This insures that, in the limit as the size of the problem goes to infinity, the error in the approximation goes to zero.

Unless the solution functions in U are at least continuous, or have some special structure, the error bound in case b of assumption 1 cannot be made arbitrarily small. Thus, by assumption 2, the size of the problems we are interested in solving will be bounded from above. If we assume that all functions in U are at least continuous, then this upper bound no longer exists. We prove this in the following lemma.

Lemma 3.1 *Assume that all $u \in U$ are continuous on the compact domain Ω. Then, for each $\epsilon > 0$, there exists a problem instance of finite size whose solution approximates the solution of the PDE to within this error tolerance for all $u \in U$.*

Proof: Assume that $u \in U$. Since Ω is compact, u is uniformly continuous on this domain. Therefore, there exists a δ such that

$$|u(\bar{z}) - u(\bar{z}')| \leq \epsilon/2$$

for all $\bar{z}, \bar{z}' \in \Omega$ for which $\|\bar{z} - \bar{z}'\|_2 \leq \delta$. And, since Ω is compact, there exists a finite set of locations $Z_\epsilon \subset \Omega$ such that

$$\Omega \subset \bigcup_{\bar{z}_j \in Z_\epsilon} B(\bar{z}_j; \delta) \ . \tag{3.1}$$

Therefore, only a finite number of solution values are required to approximate the solution anywhere in the domain to within an error tolerance of $\epsilon/2$.

By corollary 2.10, we know that there exists a finite complexity serial algorithm a that approximates the solution at each location in Z_ϵ to within an error tolerance of $\epsilon/2$. Denote the approximation to $u(\bar{z}_j)$ by $\tilde{u}_a(\bar{z}_j)$. Thus, for any $\bar{z} \in \Omega$, there exists some $\bar{z}_j \in Z_\epsilon$ such that

$$\begin{aligned} |u(\bar{z}) - \tilde{u}_a(\bar{z}_j)| &\leq |u(\bar{z}) - u(\bar{z}_j)| + |u(\bar{z}_j) - \tilde{u}_a(\bar{z}_j)| \\ &\leq \epsilon/2 + \epsilon/2 \ . \end{aligned}$$

This follows from equation 3.1. Therefore, there exists an algorithm with a finite serial complexity whose solution approximates the solution of the PDE to within an error bound of ϵ. This proves the Lemma for both cases a and b of assumption 1. ∎

The complexity of the algorithm used to prove corollary 2.10 was bounded by making some simplifying assumptions. For each $\bar{z}_j \in Z_\epsilon$, the algorithm a has the form

$$\sum_{i=1}^{\bar{l}} \sum_{l=1}^{2^{(\nu_{a,i}+P_i)\cdot d_i}} \sum_{k=1}^{m_i^{d_i}} r_{a,i,l,k}(\bar{z}_j) \cdot g_i(\bar{x}_{i,l,k}) \quad .$$

We assumed that the cost of calculating the coefficients $\{r_{a,i,l,k}(\bar{z}_j)\}$ is zero. But Lemma 3.1 will hold as long as the cost of approximating each coefficient to within an error tolerance of ϵ is finite for each $\epsilon > 0$. In general, any of the traditional finite difference and finite element techniques can be used to generate finite complexity algorithms that satisfy the same error criterion.

3.3   Architecture Independent Bounds

The next theorem essentially says the following. If the problem cannot be solved exactly using a finite amount of information, then the parallel complexity must increase as the problem size grows.

Theorem 3.1 *Assume that there exists a $\bar{z}_* \in \Omega$ and some $i \in \{1, \ldots, \bar{l}\}$ such that $d_i \neq 0$ and the following property holds: There exists a closed ball $\bar{B}(\bar{x}_*; \delta) \subset I_{d_i}$ on which $\Psi_i(\bar{z}_*, \bar{x})$ is nonzero and continuous, and on which $M_i(\bar{x})$ is nonzero. Also assume that $f_{(+)}$ is bounded away from zero for all permissible multiprocessors architectures. Then the following conditions hold for case b of assumption 1, and for case a when $\bar{z}_* \in Z$.*

- *As the problem size increases, a lower bound on the parallel complexity increases.*

- *In the limit as the problem size becomes infinite, the lower bound on the parallel complexity also becomes infinite.*

Proof: By assumptions 1 and 2 on scaling, as the size of the problem grows, the error in approximating the solution decreases. And the error bound goes to zero as the problem size becomes infinite.

For case a of assumption 1, the solution is approximated at some fixed set of solution locations Z. The error bound is on the error in these approximations. If $\bar{z}_* \in Z$, then Theorem 2.6 says that the parallel complexity must at least grow like

$$f_{(+)} \cdot \left(c_1 + c_2 \cdot \log_2 \left(\frac{1}{\epsilon} \right) \right)$$

for constants c_1 and c_2. As ϵ decreases, this bound grows. And, as $\epsilon \to 0$, this bound becomes infinite.

For case b, the error in approximating the entire solution function is being bounded. Assume that the error bound for a specific problem instance is ϵ. Then

$$|u(\bar{z}_*) - \tilde{u}(\bar{z}_*)| \leq \epsilon .$$

A problem instance specifies that a given finite number of solution values be approximated, and these are used to define the function $\tilde{u}(\bar{z})$. Let c be an upper bound on the number of solution values used to form the approximation $\tilde{u}(\bar{z}_*)$. By assumption, c is a constant independent of the problem size and ϵ. Let $Z_* = \{\bar{z}_{j,*}\}$ be the set of solution locations used when approximating $u(\bar{z}_*)$. Let $N(\bar{z}_{j,*})$ be the amount of data used to calculate $\tilde{u}(\bar{z}_{j,*})$. Then $\tilde{u}(\bar{z}_*)$ is a function of at least

$$\sum_{\bar{z}_{*,j} \in Z_*} N(\bar{z}_{j,*})$$

data.

By corollary 2.4, there exists a nonzero constant C such that $C \cdot \epsilon^{-d_i/m_i}$ is a lower bound on the amount of data required to approximate $u(\bar{z}_*)$ to within an error tolerance of ϵ. Therefore,

$$\sum_{\bar{z}_{*,j} \in Z_*} N(\bar{z}_{j,*}) \geq C \cdot \epsilon^{-d_i/m_i} ,$$

and

$$N(\bar{z}_{j,*}) \geq \frac{C}{c} \cdot \left(\frac{1}{\epsilon} \right)^{\frac{d_i}{m_i}}$$

for at least one $\bar{z}_{j,*} \in Z_*$. By Lemma 1.4, the parallel complexity of any algorithm satisfying this error bound on the approximation to the solution will be bounded from below by

$$f_{(+)} \cdot \left(\frac{d_i}{m_i} \cdot \log_2 \left(\frac{1}{\epsilon} \right) + \log_2 C - \log_2 c \right) .$$

As before, this bound grows as ϵ decreases, and the bound becomes infinite as ϵ goes to zero. This proves the Theorem. ∎

Since the parallel complexity is a lower bound on the parallel cost, the following corollary is an immediate consequence of Theorem 3.1.

Corollary 3.2 *When the assumptions of Theorem 3.1 are satisfied, the parallel cost of approximating a PDE increases unboundedly as the problem size increases unboundedly, independent of the size of the multiprocessor architecture.*

3.4   Effect of Communication Costs

3.4.1   Computation Bound Algorithms

By corollary 3.2, we usually can't bound the parallel cost as the size of the problem increases. But a good multiprocessor architecture won't exacerbate this increase. In this section we examine how the communication capabilities of a multiprocessor affect the parallel cost as the size of both the architecture and the problem increase.

Using the notation of Section 1.2.1, the parallel cost T_p of a parallel algorithm is bounded by

$$\max\{C_p, W_p\} \;\leq\; T_p \;\leq\; C_p + W_p \;.$$

C_p is the parallel complexity, and W_p is the communication cost. If the algorithm is computation bound, then $W_p \leq C_p$, and the parallel cost is at most twice as much as it would be on a multiprocessor with arbitrarily powerful communication capabilities. If the communication and the computation overlap, then the parallel cost can be as low as C_p.

As in Section 1.2.2, let $P_{a,*}$ be the maximum number of processors that collaborate in the calculation of a single solution value for a parallel algorithm a. For a given multiprocessor, let $r(p)$ be the radius of the subset of p processors with minimum radius. Note that $r(p)$ is a nondecreasing function of p. By Lemma 1.3, $r(P_{a,*})$ is a lower bound on W_p for algorithm a. For the rest of this section, we will describe conditions on $r(p)$ that determine whether parallel algorithms on a multiprocessor architecture can be computation bound.

3.4.2   Sufficient Conditions on $r(p)$

Let $N_{a,*}$ be the maximum amount of data used by algorithm a when calculating a single solution value. Assume that we have an upper bound on the serial complexity of computing a single solution value of the form[2]

$$f_{(+)} \cdot C_s(N_{a,*}) \ .$$

Assume further that this representation of the upper bound as a function of $N_{a,*}$ is common to some set of serial algorithms.

> *Example:* To calculate the solution using an implicit linear method, we
> solve a matrix equation $Ax = b$. A is a nonsingular $N \times N$ matrix, where
> $N \leq N_{a,*}$. A simple upper bound on the serial complexity of solving this
> linear system using Gaussian elimination is
>
> $$\left(f_{(+)} + f_{(*)} + f_{(/)}\right) \cdot N^3 \ .$$

See Atkinson [Atk78, pages 441-443]. Using this,

$$C_s(N_{a,*}) \ \leq \ \left(1 + \frac{f_{(*)} + f_{(/)}}{f_{(+)}}\right) \cdot N_{a,*}^3 \ .$$

In contrast, the explicit linear algorithm described in Section 2.3.2 has an upper bound of

$$\left(f_{(+)} + f_{(*)}\right) \cdot N_{a,*}$$

on the serial complexity of computing a single solution value. For this algorithm,

$$C_s(N_{a,*}) \ \leq \ \left(1 + \frac{f_{(*)}}{f_{(+)}}\right) \cdot N_{a,*} \ .$$

Until now we have concentrated on the cost of computing binary floating point operations, since they determine how the data dependence effects the complexity. For this analysis, we must also take into account the one unary operator we have assumed, negation. It can be included in our analysis easily enough given one simple assumption. We

---

[2]Note the change in notation. In Section 1.2.2, $C_s(N_{a,*})$ referred to a lower bound on the serial complexity.

assume that no algorithm in the given set calculates unnecessary unary negations, i.e., that all expressions of the form $-(-(f))$ have been reduced to f. Calculating unnecessary unary negations can only add to both the serial and parallel complexities, and has no redeeming properties. Given this assumption, the following lemma holds.

Lemma 3.2 *All parallel implementations of algorithms in the given set satisfy*

$$P_{a,*} \leq 3 \cdot C_s(N_{a,*}) + 1 \ .$$

Proof: The expression $f_{(+)} \cdot C_s(N_{a,*})$ is an upper bound on the serial complexity of using algorithm a to compute a single solution value. Since floating point addition is the cheapest binary floating point operation, at most $C_s(N_{a,*})$ binary floating point operations are required to calculate a single solution value. Therefore, there are at most $C_s(N_{a,*})$ intermediate results that might be negated. The $N_{a,*}$ data may also be negated in the algorithm. By Lemma 1.2, $N_{a,*} \leq C_s(N_{a,*}) + 1$. Therefore, at most

$$C_s(N_{a,*}) + N_{a,*} \ \leq \ 2 \cdot C_s(N_{a,*}) + 1$$

unary negations will be calculated, and at most $3 \cdot C_s(N_{a,*}) + 1$ floating point operations are required by the algorithm.

If a processor is collaborating in the computation of a particular solution value, then it will compute a floating point operation whose result is required by that value. Since there are no more than $3 \cdot C_s(N_{a,*}) + 1$ of these values, there can be no more than this many collaborating processors. ∎

Let P be the number of processors in a given multiprocessor. By Lemma 1.4,

$$C_p \geq f_{(+)} \cdot \max \left\{ \left\lceil \frac{C_s(N_{a,*})}{P_{a,*}} \right\rceil , \lceil \log_2 N_{a,*} \rceil \right\} \ .$$

Therefore, if

$$r(p) \ \leq \ f_{(+)} \cdot \max \left\{ \frac{C_s(N_{a,*})}{p} , \log_2 N_{a,*} \right\} \tag{3.2}$$

for all

$$p \in \{1, \dots, \min \{3 \cdot C_s(N_{a,*}) + 1 , P\}\} \ .$$

then this aspect of the communication cost will not prevent parallel implementations of algorithms in the set from being computation bound. This follows from Lemmas 1.4 and 3.2. Since $r(p)$ is a nondecreasing function of p, and since the right hand side of inequality 3.2 is a nonincreasing function of p, it is sufficient that $r(p)$ satisfy

$$r\left(\min\{3 \cdot C_s(N_{a,*}) + 1\,,\, P\}\right) \leq f_{(+)} \cdot \log_2 N_{a,*} \,. \tag{3.3}$$

Theorem 3.3 *Assume that $C_s(N_{a,*}) \leq \alpha \cdot N_{a,*}^{\gamma}$ for some set of serial algorithms, where $\infty > \gamma \geq 1$. Assume that no algorithm in this set calculates unnecessary unary negations. Assume that*

$$r(p) \leq \beta \cdot \lceil \log_2 p \rceil$$

on a given multiprocessor, where

$$\beta \leq \frac{f_{(+)}}{2 + \log_2(3\alpha) + \gamma} \,.$$

Then $r(p)$ will not prevent any parallel implementation of algorithms in this set from being computation bound on this multiprocessor.

Proof: Again, let P be the number of processors in the multiprocessor. Assume that $r(p) = \beta \log_2 p$, and that $3 \cdot C_s(N_{a,*}) + 1 \leq P$. Then

$$
\begin{aligned}
r(3 \cdot C_s(N_{a,*}) + 1) &\leq r(3\alpha \cdot N_{a,*}^{\gamma} + 1) \\
&= \beta \cdot \left\lceil \log_2 \left(3\alpha \cdot N_{a,*}^{\gamma} + 1\right) \right\rceil \\
&\leq \beta \cdot \left(\log_2(3\alpha \cdot N_{a,*}^{\gamma}) + 2\right) \\
&= \beta \cdot (\gamma \cdot \log_2 N_{a,*} + \log_2(3\alpha) + 2) \,.
\end{aligned}
$$

If $N_{a,*} = 1$, then $P_{a,*} = 1$ and $r(P_{a,*}) = 0$ for any β. Assume that $N_{a,*} \geq 2$. Therefore, $\log_2 N_{a,*} \geq 1$. By inequality 3.3, β must satisfy

$$\beta \cdot (\gamma \cdot \log_2 N_{a,*} + \log_2(3\alpha) + 2) \leq f_{(+)} \cdot \log_2 N_{a,*} \,.$$

This is equivalent to

$$\log_2 N_{a,*} \leq \frac{\log_2 N_{a,*}}{\gamma} \cdot \left(\frac{f_{(+)}}{\beta} - \frac{\log_2(3\alpha) + 2}{\log_2 N_{a,*}} \right) \,.$$

and a sufficient condition is

$$\beta \le \frac{f_{(+)}}{2 + \log_2(3\alpha) + \gamma} \quad .$$

If $P \le 3 \cdot C_s(N_{a,*}) + 1$, then

$$\beta \cdot \log_2 P \le \beta \cdot \log_2 \left(3 \cdot C_s(N_{a,*}) + 1\right) \quad .$$

Therefore, $r(P) \le f_{(+)} \cdot \log_2 N_{a,*}$ for the same value of β. ∎

This condition does not imply that parallel implementations of all such serial algorithms will be computation bound. It does not even indicate that there exist any parallel implementations that are computation bound. It simply says that the fact that p processors are collaborating in a computation does not prevent a parallel implementation from being computation bound.

Note that a polynomial type bound on $C_s(N_{a,*})$ will exist for all reasonable parallel algorithms, independent of the problem size. Most traditional methods generate algorithms for which $C_s(N_{a,*})$ is less than an expression of the form $\alpha \cdot N_{a,*}^3$ for all problem sizes. And, by corollary 2.10, we can get within a constant factor of the lower bound on the parallel complexity by using parallel algorithms for which α is finite and $\gamma = 1$.

Corollary 3.4 *For fixed α and γ, there is a finite transmission time t dependent only on α and γ such that the following property holds: If a multiprocessor is a member of one of the following architectural families,*

- *Fully Connected*

- *Star*

- *Distributed Shared Memory*

- *Hypercube*

and if each communication channel has a transmission time of t, then $r(p)$ satisfies the condition in Theorem 3.3. In these cases, $r(p)$ grows slowly enough that no set of parallel algorithms satisfying $C_s(N_{a,*}) \le \alpha \cdot N_{a,*}^\gamma$ is prevented from being computation bound by the mere fact that p processors are collaborating in a computation. This is independent of the multiprocessor and problem sizes.

Proof: For all of these architectural families, $r(p) \leq t \cdot \log_2 p$. See Section 1.1.3 for the discussion of this fact. Therefore, $r(p)$ is sufficiently small if $t \leq \beta$, where β is given in Theorem 3.3. ∎

3.4.3 Necessary Conditions on $r(p)$

Consider a multiprocessor with P processors and a given problem instance. Let $A(p)$ be the set of all parallel algorithms that solve the problem instance on this multiprocessor, and that use no more than p processors to calculate a single solution value. That is, if $a \in A(p)$, then $P_{a,*} \leq p$. For an algorithm a, let $C_p(a)$ be the parallel complexity of algorithm a. For each p, let

$$C_p(p) = \min_{a \in A(p)} C_p(a) \ .$$

Since $A(p_1) \subset A(p_2)$ when $p_1 < p_2$, $C_p(p_1) \geq C_p(p_2)$. Therefore, $C_p(P)$ is the minimum parallel complexity for algorithms that solve this problem instance on this multiprocessor.

Lemma 3.3 *A lower bound on the parallel cost in solving the problem instance on this multiprocessor is*

$$\min_{p \in \{1,...,P\}} \max \left\{ r(p), C_p(p) \right\} \ .$$

Proof: Assume that

$$\min_{p \in \{1,...,p'\}} \max \left\{ r(p), C_p(p) \right\} \tag{3.4}$$

is a lower bound on the parallel cost for all algorithms in $A(p')$, but assume that

$$\min_{p \in \{1,...,p'+1\}} \max \left\{ r(p), C_p(p) \right\} \tag{3.5}$$

is not a lower bound on the parallel cost for all algorithms in $A(p'+1)$. In particular, let $\hat{a}$ be some algorithm in $A(p'+1)$ whose parallel cost is less than expression 3.5.

By Lemma 1.3, $\max \left\{ r(p'+1), C_p(p'+1) \right\}$ is a lower bound on the parallel cost of all algorithms $a \in A(P)$ for which $N_{a,*} = p'+1$. Therefore, $\hat{a} \in A(p')$. But, by assumption, the parallel cost of $\hat{a}$ is no smaller than expression 3.4, which is itself no smaller than expression 3.5. This is a contradiction. Therefore, expression 3.5 is a lower bound on the

parallel complexity of all algorithms in $A(p'+1)$. Since $r(1) = 0$, expression 3.4 is always a lower bound when $p' = 1$. The Lemma then follows from an induction argument on p'.

∎

By the results in the previous section, we know that a logarithmic bound on $r(p)$ is usually sufficient to permit computation bound parallel algorithms to exist when p processors are collaborating. For now, assume that $r(p) \geq \beta \cdot p^\mu$ for this multiprocessor. For example, if the multiprocessor is a d dimensional array, then $r(p) \geq (dt/2) \cdot (p^{1/d} - 1)$. If $\beta = dt/4$ and $\mu = 1/d$, then $r(p) \geq \beta \cdot p^\mu$ when $p^\mu \geq 2$.

Let ϵ be the error tolerance specified by the problem instance. Let Z be the set of solution values specified by the problem instance. Let

$$N_{\epsilon,*} = \max_{\bar{z} \in Z} \sum_{i=1}^{\bar{l}} N_{\epsilon,i}(\bar{z}) \quad .$$

Lemma 3.4 *If $r(p) \geq \beta \cdot p^\mu$, then a lower bound on the parallel cost in solving the problem instance on this multiprocessor is*

$$\beta^{\frac{1}{\mu+1}} \cdot \left(f_{(+)} \cdot (N_{\epsilon,*} - 1) \right)^{\frac{\mu}{\mu+1}} \quad .$$

Proof: By definition, $N_{\epsilon,*}$ is a lower bound on $N_{a,*}$ for all $a \in A(P)$. Thus, by Lemma 1.4,

$$f_{(+)} \cdot \frac{N_{\epsilon,*} - 1}{p}$$

is a lower bound on $C_p(p)$ for all p. If $r(p) \geq \beta \cdot p^\mu$, then the bound in Lemma 3.3 is itself bounded from below by

$$\min_{p \in \{1,\dots,\infty\}} \max \left\{ \beta \cdot p^\mu, f_{(+)} \cdot \frac{N_{\epsilon,*} - 1}{p} \right\} \quad .$$

One term in this expression is monotonically increasing as a function of p, and one is monotonically decreasing. The minimum occurs at the value of p for which the two terms are equal,

$$p = \left(\frac{f_{(+)}}{\beta} \cdot (N_{\epsilon,*} - 1) \right)^{\frac{1}{1+\mu}} \quad .$$

Setting p equal to this value in either of the two terms gives the lower bound in the statement of the Lemma. ∎

If P is large enough, then the Green's function method described in Section 2.3.2 can be used to describe an upper bound on $C_p(p)$ for all p.

Lemma 3.5 Let $N_{gfm,*}$ be the maximum amount of data used by the algorithm described in Section 2.3.2 when calculating a single solution value. If $P \geq p \cdot |Z|$, then

$$C_p(p) \leq (f_{(*)} + f_{(+)}) \cdot \left\lceil \frac{N_{gfm,*}}{p} \right\rceil + f_{(+)} \cdot \lceil \log_2 p \rceil$$

for $p \leq N_{gfm,*}$.

Proof: For each solution value, the Green's function method can be represented by the expression

$$\sum_{i=1}^{I} \sum_{l=1}^{2^{(\nu_i + P_i) \cdot d_i}} \sum_{k=1}^{m_i^{d_i}} r_{i,l,k}(\bar{z}_j) \cdot g_i(\bar{x}_{l,k}) \ ,$$

where

$$\sum_{i=1}^{\bar{I}} 2^{(\nu_i + P_i) \cdot d_i} \cdot m_i^{d_i} \leq N_{gfm,*} \ .$$

The multiplications are all independent, and p processors can execute them in time less than or equal to

$$f_{(*)} \cdot \left\lceil \frac{N_{gfm,*}}{p} \right\rceil \tag{3.6}$$

once each processor has received the relevant data. The next stage is the summation of these products. Partition the summands into p subsets of approximately equal size, and distribute them among the processors. Summing the elements of each subset is independent, and each processor can calculate the sum of its subset in time no greater than

$$f_{(+)} \cdot \left\lceil \frac{N_{gfm,*}}{2p} \right\rceil \ . \tag{3.7}$$

This leaves at most p partial results to be summed. As in the example on page 10, this final summation can be scheduled so that the parallel complexity is at most

$$f_{(+)} \cdot \log_2 p \ . \tag{3.8}$$

If at most p processors collaborate in the calculation of a single solution value, then this algorithm is a member of $A(p)$. Thus, the sum of expressions 3.6, 3.7, and 3.8 is an upper bound on $C_p(p)$ for each p. Note that at most $N_{gfm,*}$ floating point operations can be calculated in parallel during the computation of a solution value. Therefore, this upper bound is only interesting for $p \leq N_{gfm,*}$. ∎

We will refer to the parallel algorithms in $A(P)$ that have the minimum parallel cost as *minimum cost parallel algorithms*. Note that the optimal parallel algorithms in Chapter 2 were minimum cost parallel algorithms for the ideal multiprocessor described there. Lemma 3.4 describes a lower bound on the parallel cost. Lemma 3.5 describes an upper bound on $C_p(P)$ when P is large enough. If we parallelize the Green's function method by equidistributing the processors among the different solution value computations, then at least $\lfloor P/|Z| \rfloor$ processors will be collaborating. Let $P' = \lfloor P/|Z| \rfloor$. By Lemmas 3.4 and 3.5, if

$$\beta^{\frac{1}{\mu+1}} \cdot \left(f_{(+)} \cdot (N_{\epsilon,*} - 1) \right)^{\frac{\mu}{\mu+1}}$$
$$> 2 \cdot \min_{p \in \{1,\ldots,P'\}} \left((f_{(*)} + f_{(+)}) \cdot \left\lceil \frac{N_{gfm,*}}{p} \right\rceil + f_{(+)} \cdot \lceil \log_2 p \rceil \right) \quad , (3.9)$$

then the minimum parallel cost is more than twice the minimum parallel complexity. In this case, either the minimum cost parallel algorithm is not computation bound, or the minimum cost parallel algorithm does not have the minimum parallel complexity. In both situations, the communication cost has become the dominant factor in determining the behavior of minimum cost parallel algorithms.

Theorem 3.5 *Assume that there exists a $\bar{z}_* \in \Omega$ and some $s \in \{1,\ldots,\bar{l}\}$ such that $d_s \neq 0$ and the following property holds: There exists a closed ball $\bar{B}(\bar{x}_*; \delta) \subset I_{d_s}$ on which $\Psi_s(\bar{z}, \bar{x})$ is nonzero and continuous, and on which $M_s(\bar{x})$ is nonzero. Assume that β and μ are positive constants, and that $r(p) \geq \beta \cdot p^\mu$ for all scalings of a given multiprocessor. Then the following conditions hold for case b of assumption 1, and for case a when $\bar{z}_* \in Z$.*

    1) There exists a problem instance and a multiprocessor instance such that the minimum parallel cost is greater than twice the minimum parallel complexity.

> *This condition holds for all scalings of the multiprocessor larger than the given instance.*

2) *Let $R(p)$ be the ratio of the minimum parallel cost to the minimum parallel complexity on a multiprocessor instance of size p. In the limit as the size of the problem goes to infinity, $\max_{p \geq 1} R(p)$ also goes to infinity.*

Proof: Consider case a of assumption 1 first, and assume that $\bar{z}_* \in Z$. By Theorem 2.2, there exists a constant c_1 such that

$$N_{\epsilon,*} \geq N_{\epsilon,i}(\bar{z}_*) \geq c_1 \cdot \epsilon^{-\frac{d_*}{m_*}} .$$

By Lemma 3.4, a lower bound on the minimum parallel cost is

$$\beta^{\frac{1}{\mu+1}} \cdot \left(f_+ \cdot \left(c_1 \cdot \epsilon^{-\frac{d_*}{m_*}} - 1 \right) \right)^{\frac{\mu}{\mu+1}} .$$

This is bounded from below by

$$\beta^{\frac{1}{\mu+1}} \cdot \left(\frac{f_+}{2} \cdot c_1 \right)^{\frac{\mu}{\mu+1}} \cdot \epsilon^{-\frac{d_*}{m_*} \cdot \frac{\mu}{\mu+1}}$$

when $c_1 \epsilon^{-\frac{d_*}{m_*}} \geq 2$.

Let $N_{\text{gfm},i}(\bar{z})$ be the amount of ith component data used by the Green's function method from Section 2.3.2 to approximate $u(\bar{z})$. For this algorithm, the error in approximating the ith component is less than $\epsilon/(\bar{l}+1)$. By Theorem 2.3,

$$N_{\text{gfm},*} \leq \max_{\bar{z} \in \Omega} \sum_{i=1}^{\bar{l}} N_{\text{gfm},i}(\bar{z}) \leq \max_{\bar{z} \in \Omega} \sum_{i=1}^{\bar{l}} \left(c_{2,i}(\bar{z}) \cdot \left(\frac{1}{\epsilon} \right)^{\frac{d_*}{m_*}} + 1 \right)$$

for functions $c_{2,i}(\bar{z})$ that are bounded on Ω. Therefore, there exists a finite constant c_2 such that

$$N_{\text{gfm},*} \leq c_2 \cdot \max_{i \in \{1,\dots,l\}} \left(\frac{1}{\epsilon} \right)^{\frac{d_*}{m_*}}$$

when $\epsilon < 1$. If $P = |Z| \cdot N_{\text{gfm},*}$, then

$$C_p(P) \leq C_p(N_{\text{gfm},*}) \leq (f_{(*)} + f_{(+)}) + f_{(+)} \cdot \log_2(N_{\text{gfm},*}) .$$

This follows from Lemma 3.5. Therefore,

$$C_p(P) \leq (f_{(*)} + f_{(+)}) + f_{(+)} \cdot \left(\log_2 c_2 + \max_{i \in \{1,\dots,\bar{l}\}} \frac{d_i}{m_i} \cdot \log_2 \epsilon^{-1} \right) .$$

The lower bound on the parallel cost grows linearly in $\epsilon^{-\frac{d_*}{m_*} \cdot \frac{\mu}{\mu+1}}$ as $\epsilon \to 0$, while the upper bound on $C_p(|Z| \cdot N_{gfm,*})$ only grows logarithmically in ϵ^{-1}. Thus, there exists some ϵ_* such that the parallel cost is more than twice $C_p(|Z| \cdot N_{gfm,*})$ when $\epsilon < \epsilon_*$. As $\epsilon \to 0$, the ratio of the minimum parallel cost to the minimum parallel complexity is bounded from below by

$$c_3 \cdot \frac{\epsilon^{-\frac{d_*}{m_*} \cdot \frac{\mu}{\mu+1}}}{\log_2 \epsilon^{-1}}$$

for some positive constant c_3. This can be made arbitrarily large for small enough ϵ. This has two implications.

First, for small ϵ, the minimum parallel cost will be larger than twice the minimum parallel complexity even when fewer than $|Z| \cdot N_{gfm,*}$ processors are available. For example, the upper bound on $C_p(N_{gfm,*}/2)$ is at most double the bound on $C_p(N_{gfm,*})$ for a fixed ϵ. In general, for fixed $\epsilon < \epsilon_*$, this analysis can be used to calculate a P_ϵ such that the following property holds: If the size of the multiprocessor instance is greater than P_ϵ, then the minimum parallel cost is more than twice the minimum parallel complexity.

Second, let $R(p)$ be the ratio of the minimum parallel cost to the minimum parallel complexity on a multiprocessor instance of size p. Since

$$\max_{p>1} R(p) \geq R(|Z| \cdot N_{gfm,*}) ,$$

we know that $\max_{p>1} R(p) \to \infty$ when $\epsilon \to 0$.

This proves the Theorem for case a. Case b is exactly the same, except that the lower bound on $N_{\epsilon,*}$ is now a factor of $1/c$ smaller. This bound is derived in exactly the same way as in Theorem 3.1 ∎

In summary, if $r(p) \geq \beta \cdot p^\mu$ for any positive β and μ, then the communication costs will eventually affect the achievable parallel performance. This condition holds for some common architectures.

Corollary 3.6 *Assume that the graph of a multiprocessor is a d dimensional array, and that the transmission time t is bounded away from zero. Assume that the PDE*

satisfies the conditions in Theorem 3.5. Then, for large problem sizes, the communication cost determines the minimum parallel cost as the multiprocessor is scaled up.

Proof: As before. $r(p) \geq (dt/2) \cdot (p^{1/d} - 1)$ for a d dimensional array. Therefore, if $p^{1/d} \geq 2$, then $r(p) \geq (dt/4) \cdot p^{1/d}$. The result then follows immediately from Theorem 3.5. ∎

As we mentioned in Chapter 1, the communication capabilities of large multiprocessors are constrained by the three dimensionality of the physical world, and by the speed of light. All multiprocessors can be modelled by a 3-dimensional array of processors with some finite transmission speed. This motivates the next corollary.

Corollary 3.7 *Assume that each processor is a cube with a fixed finite volume. Assume that the PDE satisfies the conditions in Theorem 3.5. Then, for large problem sizes, the communication cost determines the minimum parallel cost as the multiprocessor is scaled up.*

Proof: Assume that the volume of a processor is v in some standard unit. Then a set of p processors will take up a volume of at least pv, and cover a region whose maximum width is at least $(pv)^{1/3}$. Any message between processors must travel from a surface of the sending processor to a surface of the receiving processor. Let the physical diameter of the set be the maximum distance between two surfaces of processors in the set. Then the physical diameter is greater than

$$(pv)^{1/3} - 2v^{1/3} = (p-8)^{1/3} \cdot v^{1/3} \quad .$$

As in Section 1.1.2, let the *center* processor be the one that minimizes the maximum distance between itself and all of the others, where distance is now measured between closest surfaces. Let the physical radius be the maximum distance between the center and the other processors. Then half the physical diameter, minus half the width of the center processor, is a lower bound on the physical radius, i.e.

$$\frac{(p-8)^{1/3} \cdot v^{1/3} - v^{1/3}}{2} \quad .$$

Since no message can travel faster than the speed of light, the radius of this set of
processors is at least

$$\frac{\left((p-8)^{1/3}-1\right)\cdot v^{1/3}}{2c},$$

where c is the speed of light in these units. If $\beta = v^{1/3}/4c$ and $p > 64$, then $r(p) \geq$
$\beta \cdot p^{1/3}$. The result then follows from Theorem 3.5. ∎

Using current technologies, the speed of light is not the only restriction on transmission
speed, and $r(p) \geq \beta \cdot p^{1/3}$ for β that is much larger than that calculated in this corollary.

3.4.4   Examples

The results of Section 3.4.3 described some effects of $r(p)$ on the parallel cost of minimum
cost parallel algorithms. Similar results can also be derived for other algorithms with similar
properties. Assume that the serial complexity of computing a single solution value $u(\bar{z}_*)$
using an algorithm a is

$$\alpha \cdot f_{(+)} \cdot N_{a,*}^{\gamma}\ .$$

Here, $N_{a,*}$ is the maximum amount of data used in the calculation of a single solution
value. For example, a might be based on an implicit linear method or a Green's function
method, as in the examples on page 71. For the Green's function method, $\gamma = 1$. For an
implicit linear method, γ will normally be found in the interval $[1, 3]$.

A lower bound on the parallel cost of a parallel implementation of a is

$$\max\left\{r(p)\,,\ \frac{\alpha \cdot f_{(+)} \cdot N_{a,*}^{\gamma}}{p}\,,\ f_{(+)} \cdot \log_2 N_{a,*}\right\} \tag{3.10}$$

when p processors collaborate in approximating $u(\bar{z}_*)$. The first term is a lower bound on
the communication cost, and the second and third terms are lower bounds on the parallel
complexity. For the ideal multiprocessor of Chapter 2, $r(p) = 0$. For a hypercube based
architecture, $r(p) \geq t \cdot \log_2 p$. For a d dimensional array, $r(p) \geq (dt/2) \cdot (p^{1/d} - 1)$. Note
that this lower bound on the parallel cost is the same for the ideal multiprocessor and the
hypercube when $t = f_{(+)}$.

Assume that $t = f_{(+)}$, $\alpha = 1$, and $N_{a,*} = 1000$. Figure 3.1 on page 83 and 3.2 on
page 84 are graphs of the lower bound on the parallel cost for a variety of architectures

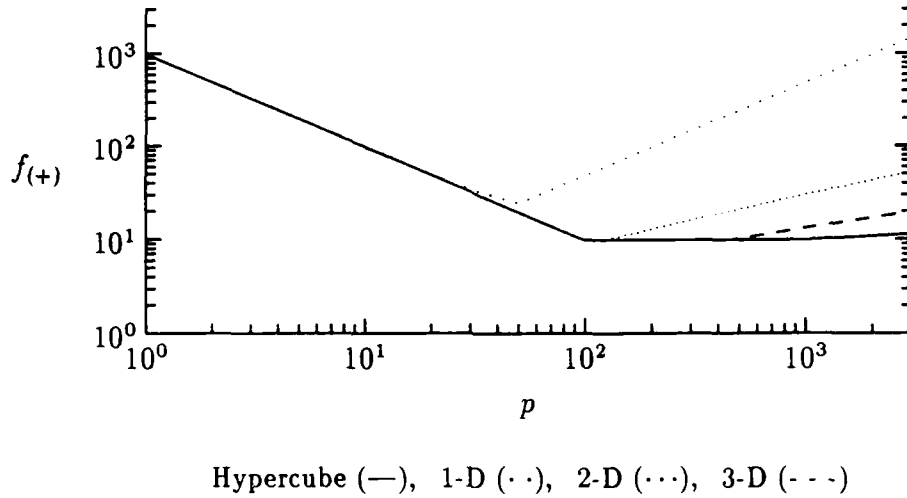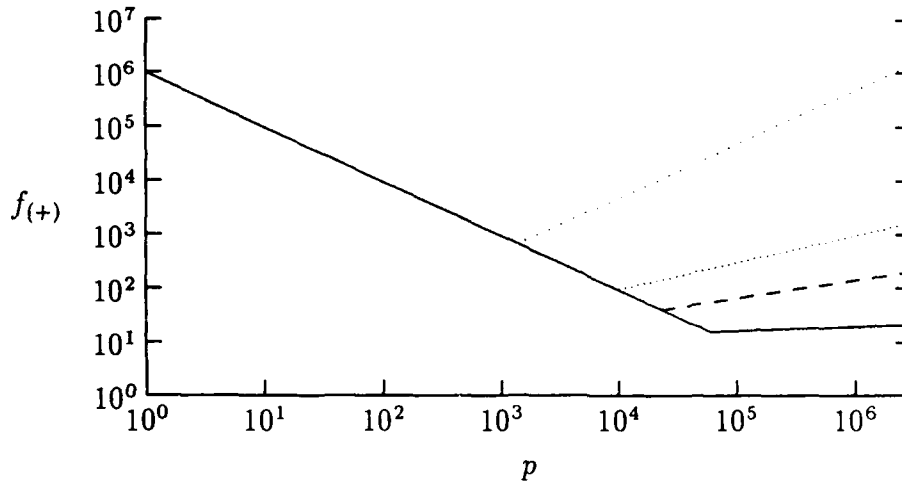Hypercube (—), 1-D (· ·), 2-D (· · ·), 3-D (- - -)

Figure 3.1: Lower bound on parallel cost as a function of the number of collaborating processors when $N_{a,*} = 1000$ and $\gamma = 1$.

and two values of γ. Each figure is a graph of the the lower bound as a function of p, where the bound is expressed in units of $f_{(+)}$. The solid curve represents a hypercube based architecture, the dashed curve represents a 1-dimensional array, the dot-dash curve represents a 2-dimensional array, and the dotted curve represents a 3-dimensional array. Figure 3.1 assumes that $\gamma = 1$, while Figure 3.2 assumes that $\gamma = 2$. From Lemma 3.2, we know that the maximum number of processors that can be used to calculate $u(\bar{z}_*)$ is bounded from above by $3 \cdot N_{a,*}^\gamma + 1$. This limit bounds the range of number of processors used in the two graphs.

In both figures the component of the lower bound representing the communication cost is the dominant term for the d dimensional arrays when p is large. And, after this term begins to dominate, the lower bound is an increasing function of p. Different assumptions about α and t will change these curves, but the general behavior of the curves will be the same if $N_{a,*}$ is large enough.

The exact behavior of the lower bound is not useful information unless the lower bound on the parallel complexity is a fairly good estimate of the true parallel complexity. Assume that the number of processors that collaborate in the approximation of $u(\bar{z}_*)$ is equal to

Hypercube (—), 1-D (· ·), 2-D (· · ·), 3-D (- - -)

Figure 3.2: Lower bound on parallel cost as a function of the number of collaborating processors when $N_{a,*} = 1000$ and $\gamma = 2$.

the maximum number of collaborating processors, $P_{a,*}$. Let P_w be the smallest number of collaborating processors for which the communication term is the dominant term in the lower bound defined in expression 3.10. Assume that the parallel complexity of algorithm a decreases monotonically as $P_{a,*}$ increases, and that

$$\frac{\alpha \cdot f_{(+)} \cdot N_{a,*}^{\gamma}}{P_{a,*}} \tag{3.11}$$

is a good estimate of the true parallel complexity when $P_{a,*} \leq P_w$. Since the parallel complexity will not increase as $P_{a,*}$ increases, and since the lower bound on the communication cost does, the parallel algorithm will not be computation bound if $P_{a,*} > P_w$. These conditions can be made to hold for the Green's function method and minimum cost parallel algorithms, and this fact led to the proof of Theorem 3.5. Most commonly used algorithms for the approximation of linear PDEs have a great deal of parallelism, and parallel implementations will exist for which expression 3.11 is a good approximation to the parallel complexity for a wide range of numbers of collaborating processors.

Note that P_w is fairly large, and that this aspect of the communication cost will not

dominate until P_w processors are used. In particular,

$$P_w = \Theta\left(N_{a,*}^{\gamma \cdot \frac{d}{d+1}}\right)$$

for a d dimensional array. Thus, for a 1-dimensional array, $P_w = \Theta\left(N_{a,*}^{\frac{1}{2}}\right)$ when $\gamma = 1$. For a 2-dimensional array, $P_w = \Theta\left(N_{a,*}^{\frac{2}{3}}\right)$ when $\gamma = 1$. For moderate sized multiprocessors and large problems, $r(p)$ will not be a limiting feature. Other aspects of communication cost may play a role though.

3.5 Conclusion

3.5.1 Summary

Any nontrivial problem instance will have a nonzero lower bound on the parallel cost that is a function of the basic processor speed. For example, if more than one datum is required to calculate some desired solution value, then Lemma 1.4 describes such a nonzero lower bound. Therefore, an infinite number of processors is unable to solve a specific problem instance arbitrarily fast unless the processor speed also becomes infinite. In this chapter we have shown that an infinite number of processors of bounded speed are also unable to bound the calculation time as the problem grows in size. This is not dependent on the multiprocessor architecture or the parallel algorithm.

Define $r(p)$ to be the minimum radius of p processor subsets of a multiprocessor architecture. Then $r(p)$ is a lower bound on the communication cost incurred when p processors collaborate in the calculation of a solution value. To prevent the multiprocessor architecture from degrading performance any further than already indicated, the growth in $r(p)$ as a function of p must be bounded. The bound on the growth of $r(p)$ is a function of the amount of parallelism in the algorithm used, but logarithmic growth is a sufficient bound. For minimum cost parallel algorithms, a necessary condition is that the bound on $r(p)$ grow slower than any positive power of p. This is not achievable in multiprocessors large enough to be constrained by the physical limitations of the three dimensional world. Therefore, the communication cost constrains the minimum parallel cost of a problem when its size becomes large.

3.5.2 Generalizations

The results of this chapter are a consequence of the following fact. The amount of data required to calculate a single solution value increases unboundedly as the size of the problem increases. The assumptions of Sections 1.3 and 3.2 are sufficient to establish this fact, but are by no means necessary. We must simply show that the error in the individual solution values must go to zero in order for the error in the approximation to go to zero, and that at least one solution value cannot be calculated exactly without an infinite amount of data. This is a common situation, and these results will hold for other reasonable assumptions as well.

If we restrict the class of algorithms more tightly, these results can be sharpened. In particular, more detailed descriptions of the effects of communication costs can be derived. For example, see Gannon and Van Rosendale [GV84]. We used the Green's function method to establish an upper bound on the minimum parallel complexity. This analysis can also be used to bound the parallel cost of parallel implementations of a Green's function method. For large numbers of processors, its minimum parallel cost is also constrained by the communication cost, unless $r(p)$ only grows logarithmically in p. The behavior of the Green's function method's parallel complexity is similar to that of more common methods, like multigrid [HT82] and particle methods [GR86], and the same type of analysis will carry over immediately.

Chapter 4

Example Bounds

In this chapter we calculate upper and lower bounds on $N_{\epsilon,i}(\bar{z})$ for some simple example problems. The bounds derived here will be somewhat tighter than those described in Chapter 2 since the generality of Theorems 2.2 and 2.3 is not necessary. We then use these results to discuss bounds on the parallel complexity and the parallel cost. We also discuss an example problem that does not satisfy some of the assumptions in Section 1.3, and describe how the analysis can be generalized for this case.

4.1 1-D Elliptic Example

Consider the problem

$$-\frac{d^2}{dx^2}u(x) = g_1(x)$$

$$u(0) = g_2, \quad u(1) = g_3$$

on the interval $[0, 1]$, for some constants g_2 and g_3. The integral representation of the solution is

$$u(z) = \int_0^1 \Psi_1(z, x)g_1(x)\, dx \ + \ g_2 \cdot (1 - z) \ + \ g_3 \cdot z \ ,$$

where

$$\Psi_1(z, x) = \begin{cases} (1 - z) \cdot x & \text{if } x \leq z, \\ (1 - x) \cdot z & \text{if } z \leq x. \end{cases}$$

There are three components,

$$u_1(z) = \int_0^1 \Psi_1(z, x) g_1(x) \, dx \quad ,$$

$$u_2(z) = g_2 \cdot (1 - z) \quad ,$$

and

$$u_3(z) = g_3 \cdot z \quad .$$

Thus, $d_1 = 1$ and $d_2 = d_3 = 0$.

Let m be a positive integer. Assume that g_1 is some member of a set G_1 that is characterized by the condition that

$$\left| \frac{d^m}{dx^m} g(x) \right| \leq m! \qquad \forall x \in [0, 1]$$

if $g \in G_1$. For example, $x^m \in G_1$. Assume that g_2 and g_3 are only known to be real numbers. That is, $G_2 = G_3 = \Re$ by the notation of Section 1.3.3.

4.1.1 Lower Bound on $N_{\epsilon,i}(.5)$

Assume that $.5 \in Z$. That is, we are interested in approximating $u(.5)$. Since $d_2 = 0$ and g_2 is not known to be bounded, the intrinsic error in approximating $u_2(.5)$ is zero if one data sample is used and infinite if none are used. Therefore, $N_{\epsilon,2}(.5) = 1$ if ϵ is finite. For the same reasons, $N_{\epsilon,3}(.5) = 1$ if ϵ is finite.

The graph of $\Psi_1(.5, x)$ as a function of x is displayed in Figure 4.1 on page 89. The function $\Psi_1(.5, x)$ is continuous and positive in any interval of the form $[\alpha, 1 - \alpha]$ when $\alpha \in (0, .5)$. Let

$$
\begin{aligned}
\epsilon(\alpha) &= C_1 \cdot \min_{x \in [\alpha, 1-\alpha]} |\Psi_1(.5, x)| \cdot m! \cdot (.5 - \alpha)^{m+1} \\
&= C_1 \cdot \frac{\alpha}{2} \cdot m! \cdot (.5 - \alpha)^{m+1} \quad ,
\end{aligned}
$$

where C_1 is the constant from Lemma 2.5. By Theorem 2.2, if $\epsilon < \epsilon(\alpha)$, then

$$
\begin{aligned}
N_{\epsilon,1}(.5) &\geq \left\lceil \left(C_1' \cdot \min_{x \in [\alpha, 1-\alpha]} |\Psi_1(.5, x)| \cdot m! \cdot (.5 - \alpha)^{m+1} \right)^{\frac{1}{m}} \cdot \left(\frac{1}{\epsilon} \right)^{\frac{1}{m}} \right\rceil \\
&= \left\lceil \left(C_1' \cdot \frac{\alpha}{2} \cdot m! \cdot (.5 - \alpha)^{m+1} \right)^{\frac{1}{m}} \cdot \left(\frac{1}{\epsilon} \right)^{\frac{1}{m}} \right\rceil \quad .
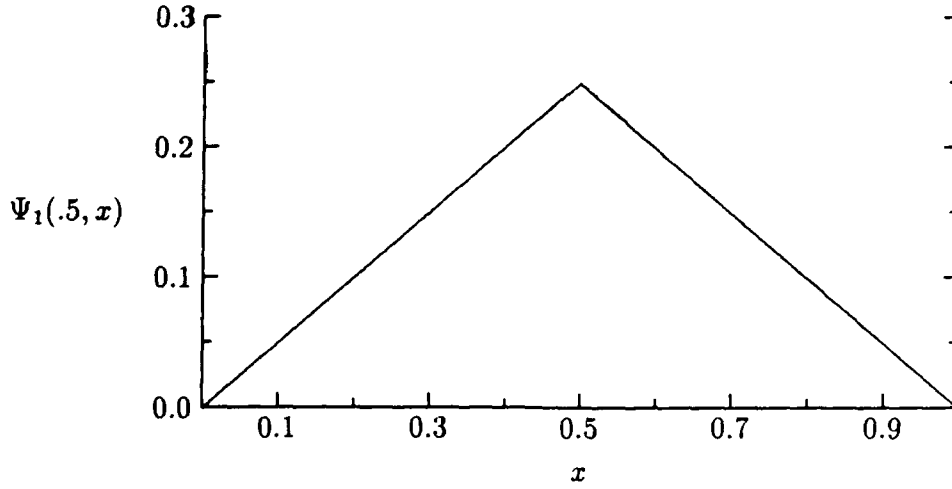\end{aligned}
\tag{4.1}
$$

Figure 4.1: $\Psi_a(.5, x)$ for $-u_{xx} = f$ given Dirichlet boundary conditions.

By the proof of Theorem 2.2, $C_1' = (1/5)^{m+1} \cdot C_1$. By the proof of Lemma 2.5 in Appendix A,

$$C_1 = \frac{1}{m!} \cdot \left(1 - \frac{1}{m+1}\right) \quad .$$

Therefore, inequality 4.1 becomes

$$N_{\epsilon,1}(.5) \geq \left\lceil \left(\left(\frac{1}{5}\right)^{m+1} \cdot \left(1 - \frac{1}{m+1}\right) \cdot \frac{\alpha}{2} \cdot (.5 - \alpha)^{m+1}\right)^{\frac{1}{m}} \cdot \left(\frac{1}{\epsilon}\right)^{\frac{1}{m}} \right\rceil \quad . \quad (4.2)$$

Inequality 4.2 can be improved. The construction of Γ on page 39 was unnecessarily conservative for the case $d_1 = 1$. A function γ of the type defined in Lemma 2.5 can be placed between every two sample locations in the interval $[\alpha, .5 - \alpha]$. This defines a new error function Γ whose support essentially covers all of $[\alpha, .5 - \alpha]$. For this construction $C_i' = C_i$, and the new lower bound is

$$N_{\epsilon,1}(.5) \geq \left\lceil \left(\left(1 - \frac{1}{m+1}\right) \cdot \frac{\alpha}{2} \cdot (.5 - \alpha)^{m+1}\right)^{\frac{1}{m}} \cdot \left(\frac{1}{\epsilon}\right)^{\frac{1}{m}} \right\rceil \quad .$$

This bound is maximized when $\alpha = 1/(2m + 4)$. The resulting lower bound on $N_{\epsilon,1}(.5)$

is

$$N_{\epsilon,1}(.5) \geq \left\lceil \left(\frac{1}{2} \cdot \frac{m+1}{m+2}\right)^{\frac{m+1}{m}} \cdot \left(\frac{m}{4 \cdot (m+1) \cdot (m+2)}\right)^{\frac{1}{m}} \cdot \left(\frac{1}{\epsilon}\right)^{\frac{1}{m}} \right\rceil \qquad (4.3)$$

when $\epsilon < \epsilon(1/(2m+4))$.

We can show directly that

$$N_{\epsilon,1}(.5) \geq \lceil m/2 \rceil \qquad (4.4)$$

if $\epsilon \geq 0$. First, assume that there are m' sampling locations, $\{x_k\}$, where $m' < \lceil m/2 \rceil$. Let $p_{2m'}(x)$ be a polynomial of the form

$$p_{2m'}(x) = K \cdot \prod_{k=1}^{m'} (x - x_k)^2 \quad .$$

Then $(d^m/dx^m) p_{2m'}(x) \equiv 0$, and $p_{2m'}$ is zero at all sampling locations. If $m > 0$, then $g_1 + p_{2m'} \in G_1$ and $g_1 + p_{2m'}$ is indistinguishable from g_1 at the sampling locations. Unlike the discussion in Section 1.3.6, the data function $g_1 + p_{2m'}$ need not be compatible with any of the other data functions. Therefore, the intrinsic error associated with a given set of sampling locations is at least as large as

$$\left| \int_0^1 \Psi_1(.5, x) p_{2m'}(x) \, dx \right| = \left| K \cdot \int_0^1 \Psi_1(.5, x) \left(\prod_{k=1}^{m'} (x - x_k)^2 \right) dx \right| \quad .$$

Since $\Psi_1(.5, x)$ is positive in $(0, 1)$, this lower bound on the intrinsic error can be made arbitrarily large by making $|K|$ arbitrarily large. Therefore, the intrinsic error is infinite if fewer than $\lceil m/2 \rceil$ sampling locations are used. This proves inequality 4.4.

Inequalities 4.3 and 4.4 together describe a lower bound on $N_{\epsilon,1}(.5)$ for all $\epsilon > 0$. We will refer to this bound by $N_{\epsilon,1}^{(L)}(.5)$. Figures 4.2 to 4.5 on pages 91 and 92 contain graphs of this lower bound as a function of ϵ for $m = 1, 2, 3,$ and 4, respectively. The constant multiplying the $\epsilon^{\frac{1}{m}}$ term in inequality 4.3 is relatively small, but $N_{\epsilon,1}(.5)$ is still guaranteed to be larger than 1000 when ϵ is less than the values in the following table.

m	1	2	3	4
$\epsilon <$	4.63×10^{-6}	2.19×10^{-9}	9.56×10^{-13}	4.16×10^{-16}

upper bound (– – –)   lower bound (···)

Figure 4.2: Bounds on $N_{\epsilon,1}(.5)$ as a function of ϵ for example 1-D elliptic problem when $m = 1$.

upper bound (– – –)   lower bound (···)

Figure 4.3: Bounds on $N_{\epsilon,1}(.5)$ as a function of ϵ for example 1-D elliptic problem when $m = 2$.
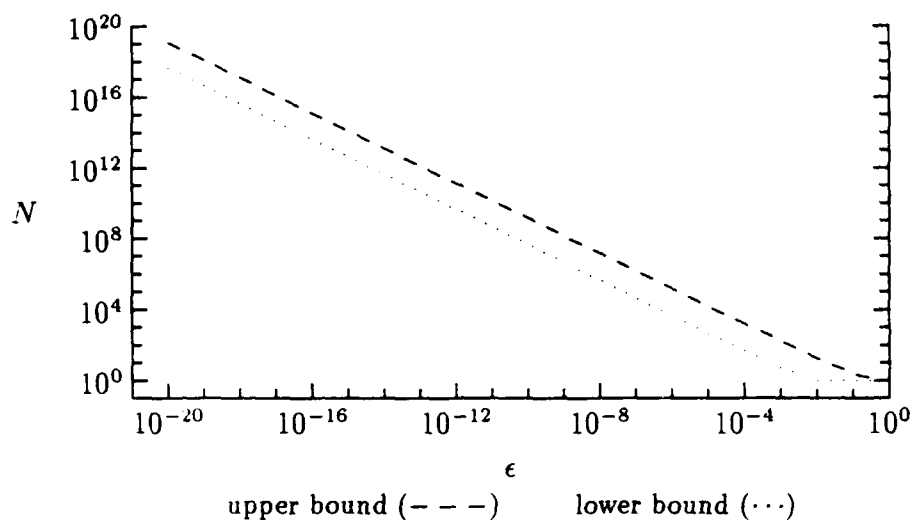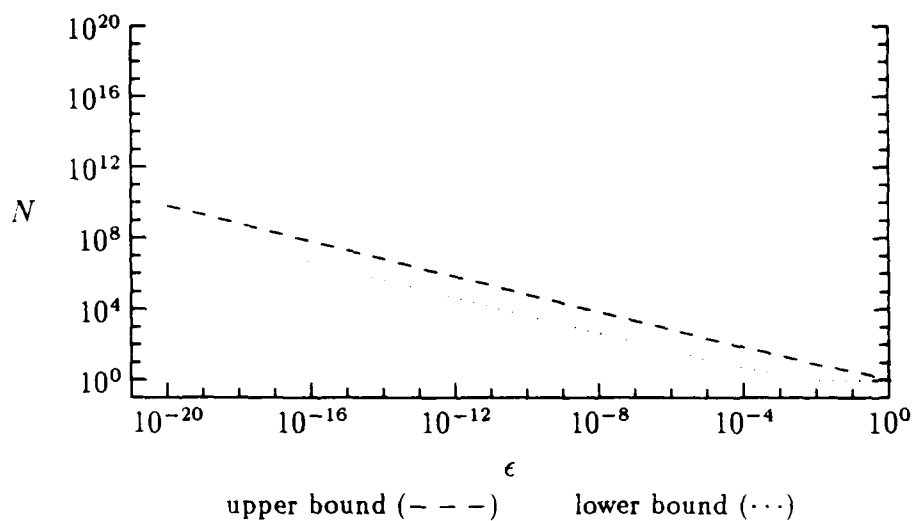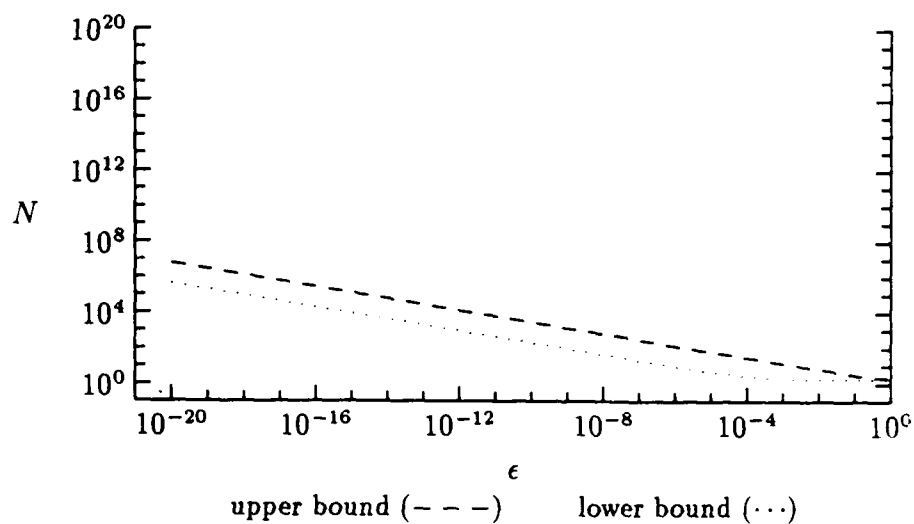
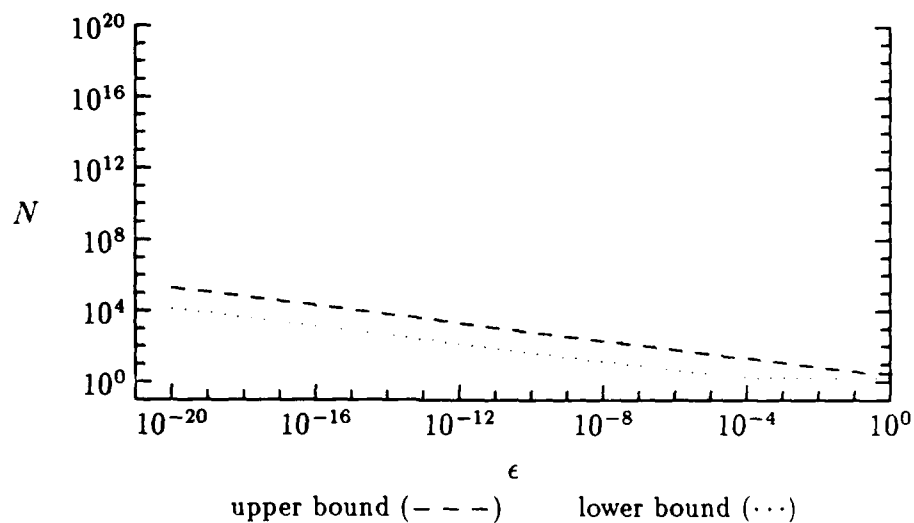Figure 4.4: Bounds on $N_{\epsilon,1}(.5)$ as a function of ϵ for example 1-D elliptic problem when $m = 3$.

Figure 4.5: Bounds on $N_{\epsilon,1}(.5)$ as a function of ϵ for example 1-D elliptic problem when $m = 4$.

If the functions in G_1 are instead characterized by the condition

$$\left|\frac{d^m}{dx^m}g(x)\right| \leq 100 \cdot m! \qquad \forall x \in [0,1] \ ,$$

then the lower bound on $N_{\epsilon,1}(.5)$ is scaled by a factor of $100^{1/m}$. Under this condition, the upper bound on ϵ guaranteeing that $N_{\epsilon,1}(.5) > 1000$ is scaled by a factor of 100.

m	1	2	3	4
$\epsilon <$	4.63×10^{-4}	2.19×10^{-7}	9.56×10^{-1}	4.16×10^{-14}

4.1.2 Upper Bound on $N_{\epsilon,1}(.5)$

For a given number of subintervals in $[0,1]$, let $\tilde{g}_1$ be the approximation to g_1 described in Section 2.2.1 on page 50. Let a be a Green's function method that approximates $u(.5)$ by

$$\tilde{u}(.5) = \int_0^1 \Psi_1(.5,x)\tilde{g}_1(x)\,dx \ + \ .5 \cdot g_2 \ + \ .5 \cdot g_3 \ .$$

Since $u_2(.5)$ and $u_3(.5)$ are computed exactly by a, an upper bound on the error in approximating $u_1(.5)$ is also an upper bound on the error in approximating $u(.5)$. Therefore, if

$$\int_0^1 |\Psi_1(.5,x)| \cdot |g_1(x) - \tilde{g}_1(x)|\,dx \ \leq \ \epsilon \ ,$$

then $a \in A_\epsilon(\{.5\})$ and $N_{\epsilon,1}(.5) \leq N_{a,0}$. Thus, we can improve the upper bound in Theorem 2.3 by modifying the argument so that ϵ is the bound on the error in the approximation to $u_1(.5)$ instead of $\epsilon/3$.

The improved upper bound on $N_{\epsilon,1}(.5)$ for this problem is

$$\begin{aligned}
N_{\epsilon,1}(.5) &\leq 2 \cdot m \cdot \left(\tilde{C}_1 \cdot \int_0^1 |\Psi_1(.5,x)|\,dx\right)^{\frac{1}{m}} \cdot (m!)^{\frac{1}{m}} \cdot \left(\frac{1}{\epsilon}\right)^{\frac{1}{m}} + 1 \\
&= 2 \cdot m \cdot \left(\frac{\tilde{C}_1}{8}\right)^{\frac{1}{m}} \cdot (m!)^{\frac{1}{m}} \cdot \left(\frac{1}{\epsilon}\right)^{\frac{1}{m}} + 1 \ ,
\end{aligned}$$

where $\tilde{C}_1$ is defined in Lemma 2.12. $\tilde{C}_1$ can be shown [Atk78, page 110] to be the maximum value of the function

$$\frac{\left|\prod_{k=1}^{m}(x - \frac{k}{m+1})\right|}{m!}$$

for $x \in [0, 1]$. This function is maximized when $x = 0$, and

$$\tilde{C}_1 \;=\; \frac{\dfrac{m!}{(m+1)^m}}{m!} \;=\; \frac{1}{(m+1)^m} \;\; .$$

Therefore,

$$N_{\epsilon,1}(.5) \;\leq\; 2 \cdot \frac{m}{m+1} \cdot \left(\frac{m!}{8}\right)^{\frac{1}{m}} \cdot \left(\frac{1}{\epsilon}\right)^{\frac{1}{m}} + 1 \;\; .$$

We will refer to this upper bound by $N_{\epsilon,1}^{(U)}(.5)$. Figures 4.2 to 4.5 on pages 91 and 4.5 also contain graphs of this upper bound on $N_{\epsilon,1}(.5)$ as a function of ϵ for $m = 1, 2, 3,$ and 4 respectively. Note that $N_{\epsilon,1}(.5)$ is no more than 1000 when ϵ is greater than the values in the following table.

m	1	2	3	4
$\epsilon >$	1.25×10^{-4}	4.45×10^{-7}	2.54×10^{-9}	1.97×10^{-11}

As before, if the functions in G_1 are characterized by the condition

$$\left| \frac{d^m}{dx^m} g(x) \right| \;\leq\; 100 \cdot m! \qquad \forall x \in [0, 1] \;\; ,$$

then the upper bound is scaled by a factor of $100^{1/m}$. Similarly, the lower bound on ϵ guaranteeing that $N_{\epsilon,1}(.5) < 1000$ is scaled by a factor of 100.

4.1.3   Comparison of Bounds

The bounds in Sections 4.1.1 and 4.1.2 are not tight. But the ratio of the upper bound $N_{\epsilon,1}^{(U)}(.5)$ to the lower bound $N_{\epsilon,1}^{(L)}(.5)$ is bounded from above by the expression

$$R(m) = 2 \cdot \frac{m}{m+1} \cdot \left(2 \cdot \frac{m+2}{m+1}\right)^{\frac{m+1}{m}} \cdot \left(\frac{(m+1) \cdot (m+2) \cdot m!}{2m}\right)^{\frac{1}{m}} + 1 \;\; .$$

for all $\epsilon > 0$. $R(m)$ is a good approximation to the ratio for small ϵ. Values of $R(m)$ are listed in the following table for $m = 1, 2, 3,$ and 4.

m	1	2	3	4
$R(m)$	28.0	15.2	14.8	15.7

Thus, the upper and lower bounds differ by approximately one order of magnitude. Note that, for large m, $R(m)$ is approximately $1.5 \cdot m$. In fact, $R(m)$ is an increasing function of m for $m \geq 3$. For example, if $m = 10$, then $R(m)$ is approximately 24.5, while if $m = 100$, then $R(m)$ is approximately 160.

4.1.4   Bounds on Parallel Complexity

By Lemma 2.1,

$$f_{(+)} \cdot \left\lceil \log_2 \left(\sum_{i=0}^{I} N_{\epsilon,i}(.5) \right) \right\rceil$$

is a lower bound on the parallel complexity of any algorithm that satisfies an error tolerance of ϵ when approximating the solution value $u(.5)$. In Section 4.1.1 we showed that $N_{\epsilon,2} = N_{\epsilon,3} = 1$ if ϵ is finite, and we calculated a lower bound on $N_{\epsilon,1}(.5)$. Therefore, by the notation of Section 4.1.1, a lower bound on the parallel complexity is

$$f_{(+)} \cdot \left\lceil \log_2 \left(2 + N_{\epsilon,1}^{(L)}(.5) \right) \right\rceil \quad . \tag{4.5}$$

In Section 4.1.2 we described an algorithm that approximated $u(.5)$ and satisfied an error tolerance ϵ. It used

$$2 + N_{\epsilon,1}^{(U)}(.5)$$

sampling locations, where

$$N_{\epsilon,1}^{(U)}(z) = 2 \cdot m \cdot \left(\tilde{C}_1 \cdot \int_0^1 |\Psi_1(z,x)|\, dx \right)^{\frac{1}{m}} \cdot (m!)^{\frac{1}{m}} \cdot \left(\frac{1}{\epsilon} \right)^{\frac{1}{m}} + 1 \quad .$$

The same approach can be used to approximate $u(z)$ for any other $z \in (0,1)$, in which case the total number of sampling locations is

$$2 + N_{\epsilon,1}^{(U)}(z) \quad .$$

As in the proof to Theorem 2.8, if an algorithm a approximates $u(z)$ in this fashion for each $z \in Z$, then the computation of each value is independent, and the parallel complexity can be as low as

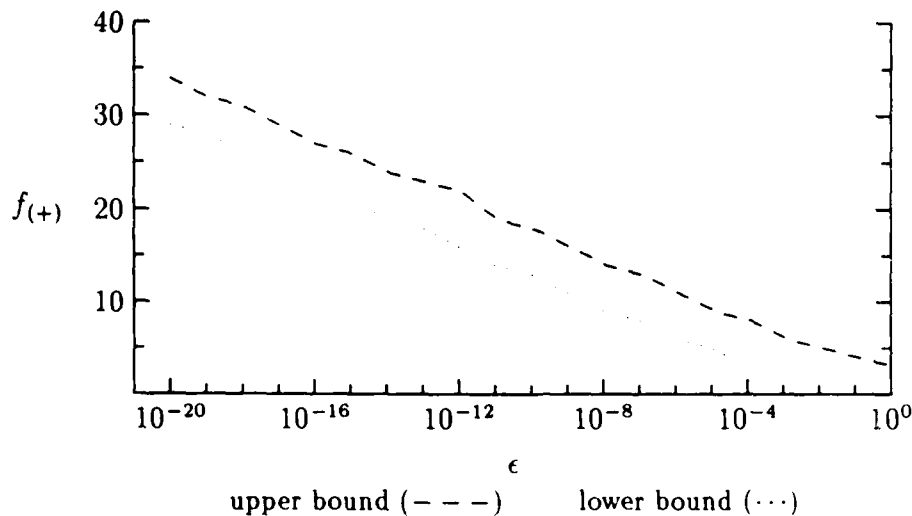$$f_{(\bullet)} + f_{(+)} \cdot \max_{z \in Z} \left\lceil \log_2 \left(2 + N_{\epsilon,1}^{(U)}(z) \right) \right\rceil \quad .$$

Figure 4.6: Bounds on minimum parallel complexity as a function of ϵ for example 1-D elliptic problem when $m = 2$.

The calculation requiring the most sampling locations is the one which maximizes the expression

$$\int_0^1 |\Psi_1(z,x)|\, dx = \frac{1}{2}(z - z^2) \ .$$

Since this is maximized when $z = .5$,

$$f_{(*)} + f_{(+)} \cdot \left\lceil \log_2(2 + N^{(U)}_{\epsilon,1}(.5)) \right\rceil \tag{4.6}$$

is an upper bound on the minimum parallel complexity of optimal parallel algorithms, independent of whether $.5 \in Z$.

Assume that $f_{(*)} = f_{(+)}$ and that $m = 2$. Then Figure 4.6 is a graph of the upper and lower bounds on the parallel complexity of optimal parallel algorithms described in inequalities 4.5 and 4.6. The parallel complexity is guaranteed to be greater than $10 \cdot f_{(+)}$ when $\epsilon \leq 10^{-9}$, while the minimum parallel complexity of an optimal parallel algorithm will be less than $10 \cdot f_{(+)}$ when $\epsilon \geq 10^{-6}$.
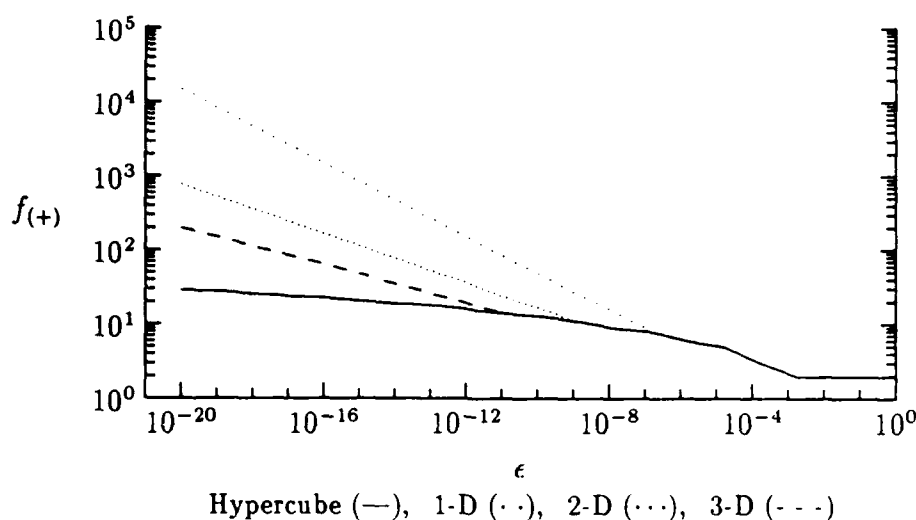
Figure 4.7: Lower bound on minimum parallel cost as a function of ϵ for example 1-D elliptic problem when $m = 2$.

4.1.5 Bounds on Parallel Cost

The bounds on the optimal parallel complexity in the previous section are very small unless ϵ is very small, but even the lower bound requires at least

$$\frac{2 + N_{\epsilon,1}^{(L)}(.5)}{2}$$

processors in order to achieve this complexity. If fewer processors collaborate in the approximation of $u(.5)$, then the bounds on the parallel complexity will be larger. Additionally, a parallel implementation on a real multiprocessor will also involve communication costs. For example, Figure 3.1 on page 83 contains graphs of lower bounds on the parallel cost as a function of the number of collaborating processors when $(2+N_{\epsilon,1}(z)) \geq 1000$. As was discussed in Section 4.1.1, if $m = 2$ and $z = .5$, then this is guaranteed to occur when $\epsilon < 2.19 \times 10^{-9}$.

By the discussion in Section 4.1.4, $2+N_{\epsilon,1}^{(L)}(.5)$ is a lower bound on the number of sampling locations required to approximate $u(.5)$ when ϵ is finite. Therefore, by corollary 1.2

on page 16,

$$\min_{p\in\{1,\dots,\infty\}} \max\left\{ r(p), \; f_{(+)} \cdot \left\lceil \frac{2 + N_{\epsilon,1}^{(L)}(.5)}{p} \right\rceil, \; f_{(+)} \cdot \left\lceil \log_2 \left(2 + N_{\epsilon,1}^{(L)}(.5) \right) \right\rceil \right\}$$

is a lower bound on the parallel cost. Assume that $m = 2$ and that $t = f_{(+)}$ for all multiprocessor architectures under consideration. Then Figure 4.7 on page 97 is a graph of this lower bound for a variety of architectures. It is clear from the Figure that the lower bound on the parallel cost is significantly larger than the lower bound on the parallel complexity when the multiprocessor is a d dimensional grid and ϵ is small.

4.2   1-D Hyperbolic Example

Consider the problem

$$\frac{\partial^2}{\partial t^2} u(x,t) - \frac{\partial^2}{\partial x^2} u(x,t) = 0 \quad \text{for} \quad (x,t) \in [0,1] \times [0,.1]$$

$$\frac{\partial}{\partial t} u(x,0) = g_1(x) \qquad \text{for} \qquad x \in [0,1]$$

$$u(x,0) = 0 \qquad \text{for} \qquad x \in [0,1]$$

$$u(0,t) = 0 = u(1,t) \qquad \text{for} \qquad t \in [0,.1] \;\;.$$

The integral representation of the solution is

$$u(z,t) = \int_0^1 \Psi_1(z,t,x) g_1(x)\, dx \;\;,$$

where

$$\Psi_1(z,t,x) = \begin{cases} \frac{1}{2} & \text{if } |z - x| \leq t \,; \\ 0 & \text{otherwise} \end{cases}$$

when $z \in (t, 1 - t)$. If $z \notin (t, 1 - t)$, then the definition of $\Psi_1(z,t,x)$ is similar, but the support is contained in an interval of length less than $2t$. Thus, there is one component, and $d_1 = 1$.

Let m be a positive integer. Assume that g_1 is some member of a set G_1 that is characterized by the condition that

$$\left| \frac{d^m}{dx^m} g(x) \right| \leq m! \qquad \forall x \in [0,1]$$

if $g \in G_1$.

4.2.1 Lower Bound on $N_{\epsilon,1}(.5,.1)$

Assume that $(.5,.1) \in Z$. The graph of $\Psi_1(.5,.1,x)$ as a function of x is displayed in Figure 4.8 on page 100. The function $\Psi_1(.5,.1,x)$ is continuous and positive for $x \in [.4,.6]$. Let

$$\epsilon_* = C_1 \cdot \min_{x \in [.4,.6]} |\Psi_1(.5,.1,x)| \cdot m! \cdot \left(\frac{.6-.4}{2}\right)^{m+1}$$

$$= \frac{m}{m+1} \cdot \frac{1}{2} \cdot \left(\frac{1}{10}\right)^{m+1} \quad .$$

By the same modification of Theorem 2.2 used in Section 4.1.1, if $\epsilon < \epsilon_*$, then

$$N_{\epsilon,1}(.5,.1) \geq \left\lceil \left(C_1' \cdot \min_{x \in [.4,.6]} |\Psi_1(.5,.1,x)| \cdot m! \cdot \left(\frac{.6-.4}{2}\right)^{m+1}\right)^{\frac{1}{m}} \cdot \left(\frac{1}{\epsilon}\right)^{\frac{1}{m}} \right\rceil$$

$$= \left\lceil \left(\frac{m}{m+1} \cdot \frac{1}{2} \cdot (.1)^{m+1}\right)^{\frac{1}{m}} \cdot \left(\frac{1}{\epsilon}\right)^{\frac{1}{m}} \right\rceil$$

$$= \left\lceil \left(\frac{m}{2(m+1)}\right)^{\frac{1}{m}} \cdot \left(\frac{1}{10}\right)^{\frac{m+1}{m}} \cdot \left(\frac{1}{\epsilon}\right)^{\frac{1}{m}} \right\rceil \qquad (4.7)$$

since $C_1' = C_1$. As in Section 4.1.1, we also have the lower bound

$$N_{\epsilon,1}(.5,.1) \geq \lceil m/2 \rceil \qquad (4.8)$$

if $\epsilon \geq 0$.

Inequalities 4.7 and 4.8 together describe a lower bound on $N_{\epsilon,1}(.5,.1)$ for all $\epsilon \geq 0$. We will refer to this bound by $N_{\epsilon,1}^{(L)}(.5,.1)$. Figure 4.9 on page 101 contains the graph of this lower bound as a function of ϵ for $m = 2$.

$N_{\epsilon,1}(.5,.1)$ is guaranteed to be larger than 1000 when ϵ is less than the values in the following table.

m	1	2	3	4
$\epsilon <$	2.50×10^{-6}	3.34×10^{-10}	3.75×10^{-14}	4.00×10^{-18}

And, as before, if the functions in G_1 are instead characterized by the condition

$$\left|\frac{d^m}{dx^m} g(x)\right| \leq 100 \cdot m! \qquad \forall x \in [0,1] \quad ,$$

then the lower bound on $N_{\epsilon,1}(.5,.1)$ is scaled by a factor of $100^{1/m}$. Similarly, the upper bound on ϵ guaranteeing that $N_{\epsilon,1}(.5,.1) > 1000$ is scaled by a factor of 100.
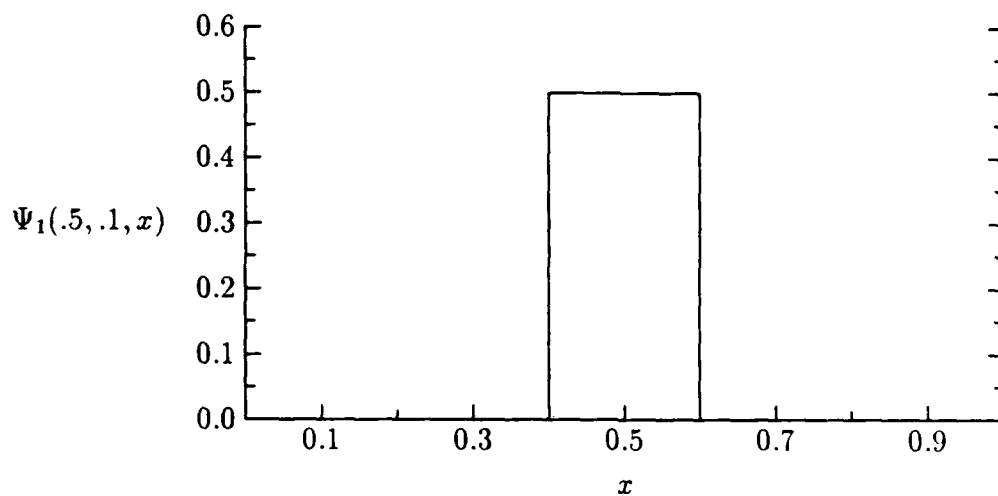
Figure 4.8: $\Psi_1(.5, .1, x)$ for one dimensional hyperbolic PDE example.

4.2.2   Upper Bound on $N_{\epsilon,1}(.5, .1)$

Since we know that $\Psi_1(.5, .1, x)$ is zero when $x \notin [.4, .6]$, we can improve on the algorithm described in Section 2.2.1. For a given number of subintervals in $[.4, .6]$, let $\tilde{g}_1$ be the approximation to g_1 described in Section 2.2.1. Let a be a Green's function method which approximates $u(.5)$ by

$$\tilde{u}(.5) = \int_{.4}^{.6} \Psi_1(.5, .1, x)\tilde{g}_1(x)\, dx \quad .$$

Then the upper bound in Theorem 2.3 can be strengthened since only one fifth as many subintervals of length $2^{-\nu_{a,1}}$ are required to cover $[.4, .6]$ as are required to cover $[0, 1]$. The improved upper bound on $N_{\epsilon,1}(.5)$ for this problem is

$$N_{\epsilon,1}(.5, .1) \leq \frac{1}{5} \cdot \left(2 \cdot m \cdot \left(\tilde{C}_1 \cdot \int_{.4}^{.6} |\Psi_1(.5, .1, x)|\, dx \right)^{\frac{1}{m}} \cdot (m!)^{\frac{1}{m}} \cdot \left(\frac{1}{\epsilon} \right)^{\frac{1}{m}} \right) + 1$$

$$= \frac{2}{5} \cdot m \cdot \left(\frac{1}{(m+1)^m} \cdot \frac{1}{10} \right)^{\frac{1}{m}} \cdot (m!)^{\frac{1}{m}} \cdot \left(\frac{1}{\epsilon} \right)^{\frac{1}{m}} + 1$$

$$= \frac{2}{5} \cdot \frac{m}{m+1} \cdot \left(\frac{m!}{10} \right)^{\frac{1}{m}} \cdot \left(\frac{1}{\epsilon} \right)^{\frac{1}{m}} + 1 \quad .$$

upper bound (– – –)   lower bound ($\cdots$)
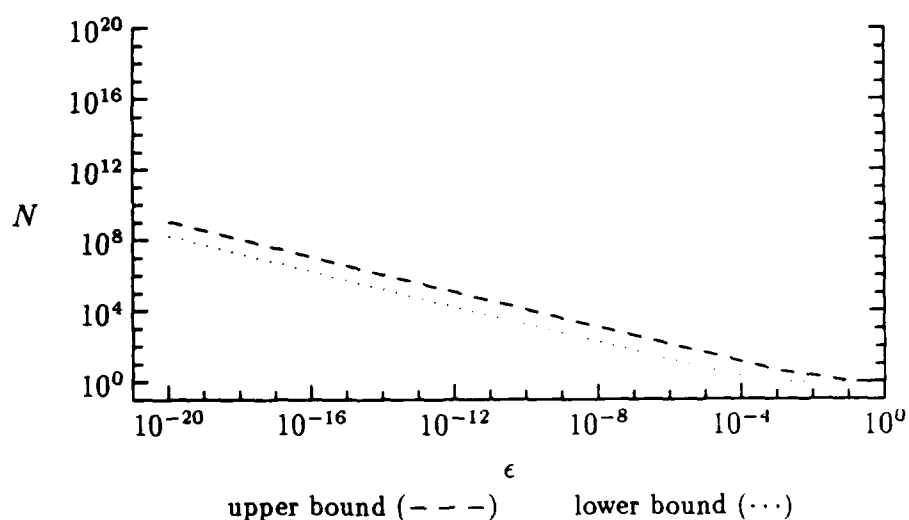
Figure 4.9: Bounds on $N_{\epsilon,1}(.5,.1)$ as a function of ϵ for example 1-D hyperbolic problem when $m = 2$.

We will refer to this upper bound by $N_{\epsilon,1}^{(U)}(.5,.1)$. Figure 4.9 also contains the graph of this upper bound on $N_{\epsilon,1}(.5,.1)$ as a function of ϵ for $m = 2$.

$N_{\epsilon,1}(.5,.1)$ is no more than 1000 when ϵ is greater than the values in the following table.

m	1	2	3	4
$\epsilon >$	2.00×10^{-5}	1.43×10^{-8}	3.25×10^{-11}	6.06×10^{-13}

If the functions in G_1 are characterized by the condition

$$\left| \frac{d^m}{dx^m} g(x) \right| \leq 100 \cdot m! \qquad \forall x \in [0,1] \ ,$$

then the upper bound is scaled by a factor of $100^{1/m}$, and the lower bound on ϵ guaranteeing that $N_{\epsilon,1}(.5,.1) < 1000$ is scaled by a factor of 100.

4.2.3   Comparison of Bounds

The ratio of the upper bound $N_{\epsilon,1}^{(U)}(.5,.1)$ to the lower bound $N_{\epsilon,1}^{(L)}(.5,.1)$ is bounded from above by the expression

$$R(m) = 4 \cdot \frac{m}{m+1} \cdot \left(2 \cdot \frac{m+1}{m}\right)^{\frac{1}{m}} \cdot (m!)^{\frac{1}{m}} + 1$$

for all $\epsilon \geq 0$. Values of $R(m)$ are listed in the following table for $m = 1, 2, 3$, and 4.

m	1	2	3	4
$R(m)$	9.00	7.53	8.56	9.91

The upper and lower bounds are a little tighter for this example than for the 1-D elliptic problem example, but they still differ by approximately one order of magnitude when $m \leq 4$. Asymptotically, $R(m)$ is again approximately $1.5 \cdot m$, and $R(m)$ is an increasing function of m for $m \geq 2$. If $m = 10$, then $R(m)$ is approximately 19, while if $m = 100$, then $R(m)$ is approximately 153.

4.2.4   Bounds on Parallel Complexity and Cost

A lower bound on the minimum parallel complexity of optimal parallel algorithms is

$$f_{(+)} \cdot \left\lceil \log_2\left(N_{\epsilon,i}^{(L)}(.5,.1)\right)\right\rceil \quad .$$

It is straightforward to show that $N_{\epsilon,1}^{(U)}(.5,.1)$ is an upper bound on $N_{\epsilon,1}(z,t)$ for all $(z,t) \in (0,1) \times [0,.1]$. Therefore, an upper bound on the minimum parallel complexity of optimal parallel algorithms is

$$f_{(*)} + f_{(+)} \cdot \left\lceil \log_2(N_{\epsilon,1}^{(U)}(.5,.1))\right\rceil \quad .$$

Assume that $f_{(*)} = f_{(+)}$ and that $m = 2$. Then the parallel complexity is guaranteed to be greater than $10 \cdot f_{(+)}$ when $\epsilon \leq 10^{-10}$, while the minimum parallel complexity of an optimal parallel algorithm will be less than $10 \cdot f_{(+)}$ when $\epsilon \geq 10^{-7}$. If fewer than $N_{\epsilon,1}(.5,.1)/2$ processors are available, or if the multiprocessor is based on a d dimensional grid, then the actual parallel cost will be significantly larger than these bounds on the minimum parallel complexity.

4.2.5   Generalization of 1-D Hyperbolic Example

Consider the problem

$$\frac{\partial^2}{\partial t^2}u(x,t) - \frac{\partial^2}{\partial x^2}u(x,t) = 0 \quad \text{for} \quad (x,t) \in [0,1] \times [0,.1]$$

$$\frac{\partial}{\partial t}u(x,0) = g_1(x) \qquad \text{for} \qquad x \in [0,1]$$

$$u(x,0) = h(x) \qquad \text{for} \qquad x \in [0,1]$$

$$u(0,t) = 0 = u(1,t) \qquad \text{for} \qquad t \in [0,.1] \ .$$

If $z \in (t, 1-t)$, then the integral representation of the solution is

$$u(z,t) = \int_0^1 \Psi_1(z,t,x)g_1(x)\,dx \ + \ \frac{1}{2}h(z-t) \ + \ \frac{1}{2}h(z+t) \ ,$$

where $\Psi_1(z,t,x)$ is the same function described in Section 4.2.1. Thus, this solution operator has three components:

$$\begin{aligned}
u_1(z,t) &= \int_0^1 \Psi_1(z,t,x)g_1(x)\,dx \ , \\
u_2(z,t) &= \frac{1}{2}h(z-t) \ , \\
u_3(z,t) &= \frac{1}{2}h(z+t) \ .
\end{aligned}$$

This solution operator does not satisfy the assumptions of Section 1.3. Components 2 and 3 are restrictions of a common function to the zero dimensional manifolds $x = z - t$ and $x = z + t$. If h is smooth, then approximate values of h can be calculated without knowing h at these two locations. For example, if ϵ is large, but finite, then $h(z)$ may be an adequate approximation to both $h(z - t)$ and $h(z + t)$.

It is still straightforward to bound the parallel complexity for this problem. Assume that h is not a linear function. Assume that $z \in (t, 1-t)$. Define $N_{\epsilon,2,3}(z,t)$ to be the minimum number of sampling locations of h in $[0,1]$ required to approximate $u_2(z,t) + u_3(z,t)$ to within an error tolerance of ϵ. The error is uncontrolled unless as least one sampling location is used. Therefore, $N_{\epsilon,2,3}(z,t) \geq 1$ for all finite $\epsilon \geq 0$. Since knowing the values $h(z - t)$ and $h(z + t)$ is sufficient,

$$N_{\epsilon,2,3}(z,t) \leq 2 \ .$$

These inequalities define upper and lower bounds on $N_{\epsilon,2,3}(z,t)$. Thus, if $(z,t) \in Z$ and $z \in (t, 1-t)$, then upper and lower bounds on the parallel complexity are

$$f_{(+)} \cdot \left\lceil \log_2 \left(1 + N_{\epsilon,1}^{(L)}(.5,.1) \right) \right\rceil$$

and

$$f_{(*)} + f_{(+)} \cdot \left\lceil \log_2 (2 + N_{\epsilon,1}^{(U)}(.5,.1)) \right\rceil \quad,$$

respectively, where $N_{\epsilon,1}^{(L)}(.5,.1)$ and $N_{\epsilon,1}^{(U)}(.5,.1)$ are defined in Sections 4.2.1 and 4.2.2. Given assumptions on h, tighter bounds can be calculated.

4.3   2-D Elliptic Example

Consider the problem

$$-\left(\frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2} \right) u(x_1, x_2) = g_1(x_1, x_2) \quad \text{for} \quad (x_1, x_2) \in B((.5,.5); .5)$$

$$u(x_1, x_2) = 0 \quad\quad\quad\quad \text{for} \quad (x_1, x_2) \in \partial B((.5,.5); .5) \quad.$$

The integral representation of the solution is

$$u(z_1, z_2) = \int_{B((.5,.5);.5)} \Psi_1(z_1, z_2, x_1, x_2) g_1(x_1, x_2) \, dx_1 \, dx_2 \quad,$$

where $\Psi_1(z_1, z_2, x_1, x_2)$ is the function

$$\frac{1}{4\pi} \cdot \log \left(\frac{\left(.25 - (z_1 - .5)(x_1 - .5) + (z_2 - .5)(x_2 - .5) \right)^2}{.25 \cdot \left((z_1 - x_1)^2 + (z_2 - x_2)^2 \right)^2} \right.$$

$$\left. + \frac{\left((z_1 - .5)(x_2 - .5) - (z_2 - .5)(x_1 - .5) \right)^2}{.25 \cdot \left((z_1 - x_1)^2 + (z_2 - x_2)^2 \right)^2} \right) \quad.$$

This can be put into the standard form by defining Ψ_1 to be zero when (z_1, z_2) or (x_1, x_2) are outside $B((.5,.5); .5)$. The data function g_1 must also be extended to the larger domain of I_2, but its values outside of $B((.5,.5); .5)$ do not influence the solution function. Thus, there is one component, and $d_1 = 2$.

Let m be a positive integer. Assume that g_1 is some member of a set G_1 that is characterized by the condition that

$$\max_{\bar{\mu} \in K_2(m)} \left| \frac{\partial^{\bar{\mu}}}{\partial \bar{x}^{\mu}} g(\bar{x}) \right| \leq m! \qquad \forall (x_1, x_2) \in [0, 1] \times [0, 1]$$

if $g \in G_1$. For example, $(x_1^m + x_2^m) \in G_1$.

4.3.1 Lower Bound on $N_{\epsilon,1}(.5, .5)$

Assume that $(.5, .5) \in Z$. If $(x_1, x_2) \in B((.5, .5); .5)$, then

$$\Psi_1(.5, .5, x_1, x_2) = \frac{1}{4\pi} \cdot \log \left(\frac{1}{4 \cdot (x_1 - .5)^2 + (x_2 - .5)^2} \right)$$

$$\equiv \Psi_1(.5, .5, r) ,$$

where $r = \sqrt{(x_1 - .5)^2 + (x_2 - .5)^2}$. The graph of $\Psi_1(.5, .5, r)$ as a function of r is displayed in Figure 4.10 on page 106. The function $\Psi_1(.5, .5, x_1, x_2)$ is continuous and positive in any subset of I_2 that excludes the location $(.5, .5)$. But the proof of Theorem 2.2 is unchanged even if the location $(.5, .5)$ is included. Therefore, consider the closed ball $\bar{B}((.5, .5); \alpha)$ for $\alpha \in (0, .5)$, and define

$$\epsilon(\alpha) = C_1 \cdot \min_{r \in [0, \alpha]} |\Psi_1(.5, .5, r)| \cdot m! \cdot \alpha^{m+2}$$

$$= C_1 \cdot \frac{1}{2\pi} \cdot \left(\log \frac{1}{2 \cdot \alpha} \right) \cdot m! \cdot \alpha^{m+2} .$$

By our generalized proof of Theorem 2.2, if $\epsilon < \epsilon(\alpha)$, then

$$N_{\epsilon,1}(.5, .5) \geq \left\lceil \left(C_1' \cdot \min_{r \in [0, \alpha]} |\Psi_1(.5, .5, r)| \cdot m! \cdot \alpha^{m+2} \right)^{\frac{2}{m}} \cdot \left(\frac{1}{\epsilon} \right)^{\frac{2}{m}} \right\rceil$$

$$= \left\lceil \left(C_1' \cdot \frac{1}{2\pi} \cdot \left(\log \frac{1}{2 \cdot \alpha} \right) \cdot m! \cdot \alpha^{m+2} \right)^{\frac{2}{m}} \cdot \left(\frac{1}{\epsilon} \right)^{\frac{2}{m}} \right\rceil . \qquad (4.9)$$

By the proof of Theorem 2.2, $C_1' = (1/5)^{m+2} \cdot C_1$. And, by the proof of Lemma 2.5 in Appendix A,

$$C_1 = \frac{\pi}{m \cdot m! \cdot m!} \cdot \left(\frac{1}{2} - \frac{1}{m+2} \right)$$

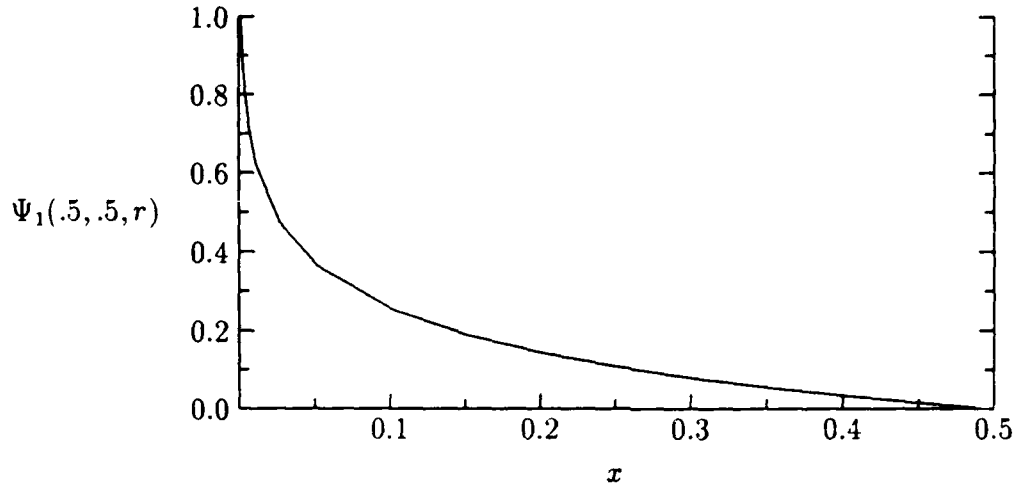$$= \frac{\pi}{m \cdot m! \cdot m!} \cdot \frac{m+1}{m+2} .$$

Figure 4.10: $\Psi_1(.5, .5, r)$ for two dimensional elliptic PDE example.

Therefore, inequality 4.9 becomes

$$N_{\epsilon,1}(.5, .5) \geq \left\lceil \left(\frac{1}{5}\right)^{\frac{2(m+2)}{m}} \cdot \left(\frac{m+1}{2 \cdot m \cdot m! \cdot (m+2)}\right)^{\frac{2}{m}} \cdot \left(\alpha^{m+2} \cdot \log\frac{1}{2 \cdot \alpha}\right)^{\frac{2}{m}} \cdot \left(\frac{1}{\epsilon}\right)^{\frac{2}{m}} \right\rceil .$$

This bound is maximized when $\alpha = .5 \cdot e^{-1/(m+2)}$. The resulting lower bound on $N_{\epsilon,1}(.5, .5)$ is

$$N_{\epsilon,1}(.5, .5) \geq \left\lceil \left(\frac{1}{10}\right)^{\frac{2(m+2)}{m}} \cdot \left(\frac{m+1}{2 \cdot e \cdot m \cdot m! \cdot (m+2)^2}\right)^{\frac{2}{m}} \cdot \left(\frac{1}{\epsilon}\right)^{\frac{2}{m}} \right\rceil \qquad (4.10)$$

when $\epsilon < \epsilon(.5 \cdot e^{-1/(m+2)})$. The same type of argument used in Section 4.1.1 proves that

$$N_{\epsilon,1}(.5, .5) \geq \lceil m/2 \rceil \qquad (4.11)$$

if $\epsilon \geq 0$.

Inequalities 4.10 and 4.11 together describe a lower bound on $N_{\epsilon,1}(.5, .5)$ for all $\epsilon > 0$. As before, we will refer to this bound by $N_{\epsilon,1}^{(L)}(.5, .5)$. Figure 4.11 on page 107 contains the graph of this lower bound as a function of ϵ for $m = 2$.

$N_{\epsilon,1}(.5, .5)$ is guaranteed to be larger than 1000 when ϵ is less than the values in the following table.

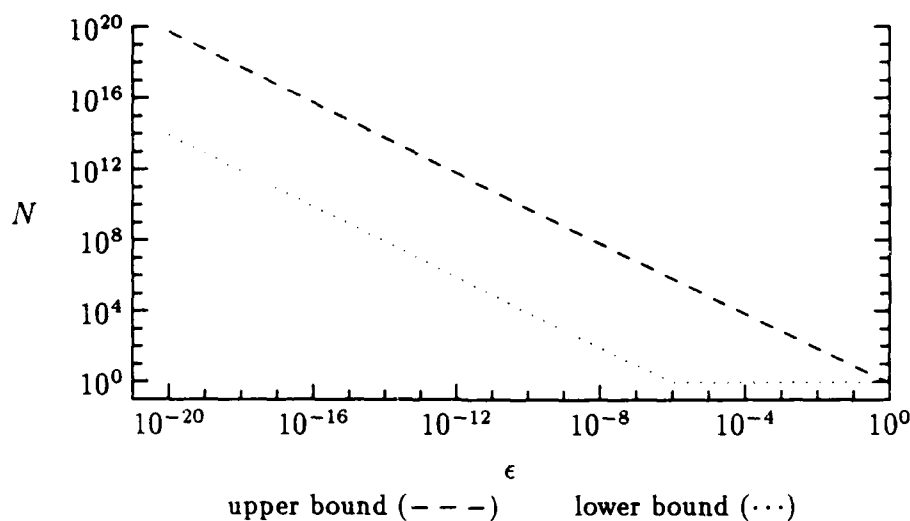upper bound (– – –)   lower bound ($\cdots$)

Figure 4.11: Bounds on $N_{\epsilon,1}(.5, .5)$ as a function of ϵ for example 2-D elliptic problem when $m = 2$.

m	1	2	3	4
$\epsilon <$	1.29×10^{-6}	8.62×10^{-10}	5.17×10^{-13}	2.66×10^{-16}

If the functions in G_1 are instead characterized by the condition

$$\max_{\bar{\mu} \in K_2(m)} \left| \frac{\partial^{\bar{\mu}}}{\partial \bar{x}^{\mu}} g(\bar{x}) \right| \leq 100 \cdot m! \qquad \forall (x_1, x_2) \in [0, 1] \times [0, 1] \ ,$$

then the lower bound on $N_{\epsilon,1}(.5, .5)$ is scaled by a factor of $100^{2/m}$. Similarly, the upper bound on ϵ guaranteeing that $N_{\epsilon,1}(.5, .5) > 1000$ is scaled by a factor of 100,

m	1	2	3	4
$\epsilon <$	1.29×10^{-4}	8.62×10^{-8}	5.17×10^{-11}	2.66×10^{-14}

4.3.2   Upper Bound on $N_{\epsilon,1}(.5, .5)$

For a given number of subcubes in $[0, 1] \times [0, 1]$, let $\tilde{g}_1$ be the approximation to g_1 described in Section 2.2.1. Let a be a Green's function method which approximates $u(.5, .5)$ by

$$\tilde{u}(.5, .5) = \int_{I_2} \Psi_1(.5, .5, x_1, x_2) \tilde{g}_1(x_1, x_2) \, dx_1 \, dx_2 \ .$$

The upper bound in Theorem 2.3 is

$$N_{\epsilon,1}(.5,.5) \;\leq\; 4 \cdot m^2 \cdot \left(\tilde{C}_1 \cdot \int_{I_2} |\Psi_1(.5,.5,x_1,x_2)| \, dx_1 \, dx_2 \right)^{\frac{2}{m}} \cdot (m!)^{\frac{2}{m}} \cdot \left(\frac{1}{\epsilon} \right)^{\frac{2}{m}} + 1$$

$$= \; 4 \cdot m^2 \cdot \left(\tilde{C}_1 \cdot \frac{1}{32} \right)^{\frac{2}{m}} \cdot (m!)^{\frac{2}{m}} \cdot \left(\frac{1}{\epsilon} \right)^{\frac{2}{m}} + 1 \ .$$

By the proof of Lemma 2.12 in Appendix B and the discussion in Section 4.1.1, we know that

$$\tilde{C}_1 = \frac{1 + 2^m}{(m+1)^m} \ .$$

Therefore

$$N_{\epsilon,1}(.5,.5) \leq 4 \cdot \left(\frac{m}{m+1} \right)^2 \cdot \left(\frac{2^m + 1}{32} \right)^{\frac{2}{m}} \cdot (m!)^{\frac{2}{m}} \cdot \left(\frac{1}{\epsilon} \right)^{\frac{2}{m}} + 1 \ .$$

We will refer to this upper bound by $N_{\epsilon,1}^{(U)}(.5,.5)$. Figure 4.9 also contains the graph of this upper bound on $N_{\epsilon,1}(.5,.5)$ as a function of ϵ for $m = 2$.

$N_{\epsilon,1}(.5,.5)$ is no more than 1000 when ϵ is greater than the values in the following table.

m	1	2	3	4
$\epsilon >$	2.97×10^{-3}	5.56×10^{-4}	1.80×10^{-4}	8.37×10^{-5}

If the functions in G_1 are characterized by the condition

$$\max_{\bar{\mu} \in K_2(m)} \left| \frac{\partial^{\bar{\mu}}}{\partial \bar{x}^{\bar{\mu}}} g(\bar{x}) \right| \;\leq\; 100 \cdot m! \qquad \forall (x_1, x_2) \in [0,1] \times [0,1] \ ,$$

then the upper bound is scaled by a factor of $100^{2/m}$, and the lower bound on ϵ guaranteeing that $N_{\epsilon,1}(.5) < 1000$ is scaled by a factor of 100.

4.3.3 Comparison of Bounds

The ratio of the upper bound $N_{\epsilon,1}^{(U)}(.5,.5)$ to the lower bound $N_{\epsilon,1}^{(L)}(.5,.5)$ is bounded from above by the expression

$$R(m) = 400 \cdot \left(\frac{25 \cdot e}{4} \cdot \frac{2^m + 1}{m+1} \cdot m \cdot (m!)^2 \cdot (m+2) \right)^{\frac{2}{m}} \cdot \left(\frac{m}{m+1} \right)^2 + 1$$

for all $\epsilon \geq 0$. Values of $R(m)$ are listed in the following table for $m = 1, 2, 3$, and 4.

m	1	2	3	4
$R(m)$	5.26×10^7	6.44×10^6	4.95×10^6	5.60×10^6

The upper and lower bounds are much worse for this example than for the one dimensional examples. They now differ by approximately seven orders of magnitude when $m \leq 4$. Techniques for improving the bounds are described in Section 4.3.5. Asymptotically, $R(m)$ is approximately $29.3 \cdot m^4$, and $R(m)$ is an increasing function of m for $m \geq 3$.

4.3.4 Bounds on Parallel Complexity and Cost

As in Section 4.2.4, a lower bound on the minimum parallel complexity of optimal parallel algorithms is

$$f_{(+)} \cdot \left\lceil \log_2 \left(N_{\epsilon,i}^{(L)}(.5, .5) \right) \right\rceil \quad .$$

It is again straightforward to show that $N_{\epsilon,1}^{(U)}(.5, .5)$ is an upper bound on $N_{\epsilon,1}(z_1, z_2)$ for all $(z_1, z_2) \in B((.5, .5); .5)$. Therefore, an upper bound on the minimum parallel complexity of optimal parallel algorithms is

$$f_{(*)} + f_{(+)} \cdot \left\lceil \log_2(N_{\epsilon,1}^{(U)}(.5, .5)) \right\rceil \quad .$$

Assume that $f_{(*)} = f_{(+)}$, and that $m = 2$. Then the parallel complexity is guaranteed to be greater than $10 \cdot f_{(+)}$ when $\epsilon \leq 10^{-10}$, while the minimum parallel complexity of an optimal parallel algorithm will be less than $10 \cdot f_{(+)}$ when $\epsilon \geq 10^{-3}$. As before, if fewer than $N_{\epsilon,1}(.5, .5)/2$ processors are available, or if the multiprocessor is based on a d dimensional grid, then the actual parallel cost will be significantly larger than these bounds on the minimum parallel complexity.

4.3.5 Improving the Bounds

The bounds calculated for this problem describe nontrivial constraints on the parallel cost of computing the solution, but the discrepancy in the upper and lower bounds is very large. There are several ways to improve these bounds. For concreteness, we will limit the discussion of the amount of improvement to the case $m = 2$.

a) $\Psi_1(z_1, z_2, x_1, x_2)$ is identically zero when (x_1, x_2) is outside the ball $B((.5, .5); .5)$, and sampling locations outside the ball are not really needed. Thus, when constructing the upper bound, g_1 only needs to be approximated in subcubes that intersect the ball. This decreases the upper bound by a factor of approximately $\pi/4$ when the subcubes are small. This is the same approach used to improve the estimates for the example 1-D hyperbolic problem.

b) The generalizations described in Sections 2.1.4 and 2.2.2 base the sampling frequency in any subregion R on the value of Ψ_1 in that subregion. If the volume of R, $\text{Vol}(R)$, is small and $\Psi_1(.5, .5, x_1, x_2)$ is continuous in R, then

$$\int_R |\Psi_1(.5, .5, x_1, x_2)| \, dx_1 \, dx_2 \approx \left(\min_{(x_1, x_2) \in R} \Psi_1(.5, .5, x_1, x_2) \right) \cdot \text{Vol}(R) \ .$$

Since Ψ_1 is a type 1 function, the upper and lower bounds on the sample size are only functions of Ψ_1 over regions where it is continuous. Thus, the upper and lower bounds will be similar functions of Ψ_1.

We can estimate the amount of improvement that is possible from using these generalizations. Assume that $\Psi_1(.5, .5, x_1, x_2) = 1$ when $(x_1, x_2) \in B((.5, .5); .5)$, and is identically zero outside the ball. Now the upper and lower bounds depend on Ψ_1 in exactly the same way. If $m = 2$, then the ratio of the bounds for this problem is approximately $1/27$ times the ratio of the bounds for the original problem.

c) The calculation of the lower bound on the number of sampling locations was very rough. In particular, in Lemma 2.11 the optimal placement of the sampling locations for this construction was simply shown to satisfy

$$r_k \geq \frac{1}{5} \cdot \frac{\delta}{n^{1/d_1}}$$

in a d dimensional ball $B(\bar{x}_*; \delta)$. Here r_k is the maximum distance between the kth sampling location and the boundary of its Voronoi cell, and n is the number of sampling locations in the ball. While the analysis described here was not sufficient to prove it, we expect that the optimal sampling locations in $B(\bar{x}_*; \delta)$ will be essentially equidistributed. In this case

$$r_k \approx \frac{\delta}{n^{1/d_1}}$$

for all k. For this type of distribution, the error function Γ can be constructed so that almost every sampling location $\bar{w}$ "contributes" a function $\gamma_{\bar{w}}$ to Γ. Lemma 2.9 can then be modified to state that the support of this new Γ essentially covers the entire ball, and the resulting lower bound is increased by a factor of 5^{m+d}. Thus, when $m = 2$, the lower bound for the example 2-D elliptic problem is increased by a factor of 625.

Improvements a–c can theoretically decrease the ratio by a factor of

$$\frac{\pi}{4} \cdot \frac{1}{27} \cdot \frac{1}{625} \approx 4.66 \times 10^{-5}$$

when $m = 2$. The ratio of the improved upper bound to the improved lower bound is approximately 300. Note that the majority of the change reflects an increase in the lower bound. Thus, for this problem, the upper bound is a much better estimate of the true value of $N_{\epsilon,1}(.5, .5)$.

To improve the bounds further requires improving the constants C_1 and $\tilde{C}_1$. For example, C_1 is primarily a function of the type of error function used in the construction of the lower bound. The error function described in Chapter 2 is very conservative for dimensions greater than or equal to 2. It is a C^∞ function, and is zero at many locations other than those required by the analysis. Thus, we expect that this constant can be increased.

4.4 Summary

The results of Chapters 2 and 3 were sufficient to establish the behavior of $N_{\epsilon,1}(\bar{z})$ as a function of ϵ, and to bound the parallel complexity and cost of minimum cost parallel algorithms. For the one dimensional example problems described in this chapter, we were able to substantially tighten these bounds by taking advantage of the specifics of the problems. But, even without using the improvements described in Section 4.3.5, we were able to calculate nontrivial bounds on the intrinsic cost of the two dimensional example problem. These bounds become even more striking as the assumptions on the function sets $\{G_i\}$ become weaker.

The problem assumptions made in Section 1.3 are not satisfied by all classes of linear PDEs. Section 4.2.5 described a problem where certain components can be approximated without using sampling locations in the domain of the component. This is typical of hyperbolic PDEs. But, as demonstrated, the analysis could be generalized by treating these components separately. While this particular example was trivial, a similar approach will work for more general cases.

Chapter 5

Conclusions

The cost of calculating a numerical approximation to the solution of a partial differential equation is constrained by the amount of data it must use in order to calculate an accurate enough approximation. For the simple assumptions about the data function and the partial differential equation described in Section 1.3, these constraints limit how quickly the approximation can be calculated. As ϵ, the bound on the error in the approximation, decreases, there exists a lower bound on the parallel complexity that must increase at least linearly as a function of

$$\log_2 \left(\frac{1}{\epsilon} \right) \ .$$

Moreover, this bound is tight in the sense that there exists a family of explicit linear algorithms with parallel implementations whose parallel complexity grows no faster than this.

The number of processors used to calculate the approximation must increase as the error bound decreases if the lower bound on the parallel complexity is to be achieved. This usually entails an increase in the communication cost. But the communication cost does not change the asymptotic behavior as long as the parallel algorithm remains computation bound. This is not always possible for small error tolerances when the approximation is calculated on a multiprocessor whose radius grows faster than logarithmically. In particular, the parallel cost of optimal algorithms on d dimensional grid architectures is determined by the communication capabilities of the architecture rather than by the minimum parallel complexity. But, for a given processor speed, there exists a logarithmic bound on the

radius of a multiprocessor that ensures that the communication cost associated with the radius will not influence the behavior of the parallel cost of optimal parallel algorithms.

As mentioned in Section 3.5.2, the assumptions used to prove these results are merely sufficient, and not necessary. We contend that they are reasonable assumptions for a priori estimates on the error for mth order accurate approximations, $m < \infty$, to the solution of many important classes of linear scalar partial differential equation. And the results can be extended to any problem with a solution operator that can be represented as a sum of integral components if at least some of the components satisfy the assumptions. A simple example of this was described in Section 4.2.5

The assumptions and analysis described in the previous chapters are also reasonable for other classes of problems.

- Very little changes for systems of linear partial differential equations. Many linear problems also have integral representations for the solution operator for each element of the solution vector, where the kernels satisfy the usual assumptions.

- For nonlinear problems the solution operator will not have such a simple representation. But most numerical approximations to nonlinear problems can be interpreted as solutions to a sequence of linear problems. Thus, the results carry over immediately if these linear subproblems satisfy the assumptions. The only difficulty is that the solution u is not necessarily smooth when the data is smooth. Thus, the number of solution values required to represent the solution function can be arbitrarily large. For most practical applications the solution is smooth almost everywhere, and discontinuities are limited to a finite number of bounded smooth lower dimensional surfaces. The additional cost is then in capturing or tracking these surfaces. This is also approximated by the calculation of a finite number of values dependent on the error bound.

- The data functions are discontinuous for many applications. But, similar to the discussion of nonlinear problems, the data is generally smooth on all but a finite number of bounded smooth lower dimensional surfaces. The analysis follows through as before as long as the solution is also smooth except on a finite number of bounded smooth lower dimensional surfaces for this type of data.

- The analysis only depends on the solution operator having the required integral formulation. Thus, the results are not restricted to linear partial differential equations.

Appendix A

Construction of the Error Function in I_{d_i}

In this appendix we prove Lemma 2.5:

Assume that $d_i > 0$. For any $\bar{x}_* \in I_{d_i}$ and $\delta > 0$ such that $B(\bar{x}_*; \delta) \subset I_{d_i}$, there exists a function $\gamma \in C^\infty(I_{d_i})$ with the following properties.

1) $\gamma \in G_i$.

2) $\gamma(\bar{x}) = 0$ for all $\bar{x} \notin B(\bar{x}_*; \delta)$.

3) $\gamma(\bar{x}) \geq 0$ for all $\bar{x} \in B(\bar{x}_*; \delta)$.

4) There exists a constant $C_i > 0$ depending only on m_i, d_i, and $\|\cdot\|_{(i)}$ such that

$$\int_{I_{d_i}} \gamma(\bar{x}) \, d\bar{x} \geq C_i \cdot \delta^{m_i + d_i} \cdot \min_{\bar{x} \in \bar{B}(\bar{x}_*; \delta)} M_i(\bar{x}) \ ,$$

where $\bar{B}(\bar{x}_*; \delta)$ is the closure of $B(\bar{x}_*; \delta)$.

We will prove the Lemma by constructing a spherically symmetric function in $B(\bar{x}_*; \delta)$ whose integral is the limit of integrals of functions that satisfy conditions 1-3. We begin by proving a condition on the radial derivative that insures membership in G_i.

117

A.1   Sufficient Bounds on Directional Derivatives

Functions in G_i are characterized by the bound

$$\|\bar{\nabla}^{(m_i)}g(\bar{x})\|_{(i)} \leq M_i(\bar{x})$$

and membership in $C^{m_i}(I_{d_i})$. A sufficient condition for g to be in G_i is described by the following lemma.

Lemma A.1 *Let g be a $C^\infty(I_{d_i})$ function with the following property. There exists a constant $\delta > 0$, a location $\bar{x}_* \in I_{d_i}$, and a function $\tilde{g}(r)$ such that*

1) $g(\bar{x}) = \tilde{g}(\|\bar{x} - \bar{x}_*\|_2)$ in the d_i dimensional open ball $B(\bar{x}_*; \delta)$.

2) $g(\bar{x}) = 0$ if $\bar{x} \notin B(\bar{x}_*; \delta)$.

Assume further that $\tilde{g}$ satisfies the condition that

$$\frac{\partial^s}{\partial r^s}\tilde{g}(0) = 0$$

for all $s \in \{1, \ldots, m_i - 1\}$. Then there exists a constant $\tilde{C}_i > 0$ depending only on $\|\cdot\|_{(i)}$, m_i, and d_i such that $g \in G_i$ when

$$\max_{r \in [0,\delta]} \left| \frac{\partial^{m_i}}{\partial r^{m_i}}\tilde{g}(r) \right| \leq \tilde{C}_i \cdot \min_{\bar{x} \in B(\bar{x}_*; \delta)} M_i(\bar{x}) \ .$$

Proof: By the equivalence of norms over finite dimensional spaces [Atk78], we know that there exist positive constants c_1 and c_2 such that

$$c_1 \cdot \|\bar{\nabla}_{d_i}^{(m_i)}g(\bar{x})\|_\infty \leq \|\bar{\nabla}_{d_i}^{(m_i)}g(\bar{x})\|_{(i)} \leq c_2 \cdot \|\bar{\nabla}_{d_i}^{(m_i)}g(\bar{x})\|_\infty \ , \qquad (A.1)$$

where $\|(x_1, \ldots, x_s)\|_\infty = \max_{l \in \{1, \ldots, s\}} |x_l|$. Therefore, if

$$\|\bar{\nabla}_{d_i}^{(m_i)}g(\bar{x})\|_\infty \leq \frac{1}{c_2} \cdot M_i(\bar{x}) \ ,$$

then

$$\|\bar{\nabla}_{d_i}^{(m_i)}g(\bar{x})\|_{(i)} \leq M_i(\bar{x}) \ .$$

Since $g \in C^{\infty}(I_{d_i})$, g is also a member of $C^{m_i}(I_{d_i})$ for any m_i. Therefore, to prove the result, it is sufficient to find a constant $\tilde{C}_i$ such that

$$\left| \frac{\partial^{\bar{\mu}}}{\partial \bar{x}^{\bar{\mu}}} g(\bar{x}) \right| \leq \frac{1}{c_2} \cdot M_i(\bar{x}) \qquad \forall \bar{\mu} \in K_{d_i}(m_i) , \; \forall \bar{x} \in I_{d_i}$$

when

$$\left| \frac{\partial^{m_i}}{\partial r^{m_i}} \tilde{g}(r) \right| \leq \tilde{C}_i \cdot \min_{\bar{x} \in B(\bar{x}_*;\delta)} M_i(\bar{x}) \qquad \forall r \in [0, \delta] \; .$$

When $d_i = 1$, it is clear that choosing $\tilde{C}_i = 1/c_2$ verifies the Lemma. For the rest of this proof, we will assume that $d_i > 1$.

If $\bar{x} \notin B(\bar{x}_*;\delta)$, then $\partial^{\bar{\mu}} g(\bar{x})/\partial \bar{x}^{\bar{\mu}} \equiv 0$ for all $\bar{\mu} \in K_{d_i}(m_i)$, and any $\tilde{C}_i > 0$ will suffice. Assume that $\bar{x} \in B(\bar{x}_*;\delta)$. Let $\bar{\mu}$ be some element of $K_{d_i}(m_i)$. By a repeated application of the chain rule,

$$\frac{\partial^{\bar{\mu}}}{\partial \bar{x}^{\bar{\mu}}} \tilde{g}(\|\bar{x} - \bar{x}_*\|_2) = \sum_{s=1}^{m_i} \alpha_{\bar{\mu},s}(\bar{x}) \frac{\partial^s}{\partial r^s} \tilde{g}(\|\bar{x} - \bar{x}_*\|_2) \tag{A.2}$$

where $|\alpha_{\bar{\mu},s}(\bar{x})| \leq m_i! \cdot (\|\bar{x} - \bar{x}_*\|_2)^{s-m_i}$. Replace each expression of the form

$$(\partial^s/r^s)\tilde{g}(\|\bar{x} - \bar{x}_*\|_2)$$

by its Taylor expansion around $r = 0$,

$$\frac{\partial^s}{\partial r^s} \tilde{g}(\|\bar{x} - \bar{x}_*\|_2) = \sum_{t=s}^{m_i - 1} \frac{(\|\bar{x} - \bar{x}_*\|_2)^{t-s}}{(t-s)!} \cdot \frac{\partial^t}{\partial r^t} \tilde{g}(0) + \frac{(\|\bar{x} - \bar{x}_*\|_2)^{m_i - s}}{(m_i - s)!} \frac{\partial^{m_i}}{\partial r^{m_i}} \tilde{g}(\xi_s)$$

$$= \frac{(\|\bar{x} - \bar{x}_*\|_2)^{m_i - s}}{(m_i - s)!} \cdot \frac{\partial^{m_i}}{\partial r^{m_i}} \tilde{g}(\xi_s)$$

where $\xi_s \in [0, \|\bar{x} - \bar{x}_*\|_2]$. Equation A.2 reduces to

$$\frac{\partial^{\bar{\mu}}}{\partial \bar{x}^{\bar{\mu}}} \tilde{g}(\|\bar{x} - \bar{x}_*\|_2) = \sum_{s=1}^{m_i} \alpha_{\bar{\mu},s}(\bar{x}) \cdot \frac{(\|\bar{x} - \bar{x}_*\|_2)^{m_i - s}}{(m_i - s)!} \cdot \frac{\partial^{m_i}}{\partial r^{m_i}} \tilde{g}(\xi_s) \; .$$

Thus,

$$\left| \frac{\partial^{\bar{\mu}}}{\partial \bar{x}^{\bar{\mu}}} g(\bar{x}) \right| = \left| \frac{\partial^{\bar{\mu}}}{\partial x^{\bar{\mu}}} \tilde{g}(\|\bar{x} - \bar{x}_*\|_2) \right| \leq m_i \cdot m_i! \cdot \max_{\xi \in [0, \|\bar{x} - \bar{x}_*\|_2]} \left| \frac{\partial^{m_i}}{\partial r^{m_i}} \tilde{g}(\xi) \right| \; ,$$

for all $\bar{\mu} \in K_{d_i}(m_i)$. Therefore, if

$$\left| \frac{\partial^{m_i}}{\partial r^{m_i}} \tilde{g}(\xi) \right| \leq \frac{1}{c_2 \cdot m_i \cdot m_i!} \cdot \min_{\bar{x} \in B(\bar{x}_*; \delta)} M_i(\bar{x}) \qquad \forall r \in [0, \delta] \ ,$$

then

$$\left| \frac{\partial^{\bar{\mu}}}{\partial \bar{x}^{\bar{\mu}}} g(\bar{x}) \right| \leq \frac{1}{c_2} \cdot M_i(\bar{x}) \qquad \forall \bar{x} \in I_{d_i} \ .$$

Thus, choosing

$$\tilde{C}_i = \frac{1}{c_2 \cdot m_i \cdot m_i!}$$

verifies the Lemma when $d_i > 1$. ∎

A.2   Error Function

Let $M_i' = \tilde{C}_i \cdot \min_{\bar{x} \in B(\bar{x}_*; \delta)} M_i(\bar{x})$, where $\tilde{C}_i$ is the constant referred to in Lemma A.1. Let $\gamma(\bar{x}) = \tilde{\gamma}(\|\bar{x} - \bar{x}_*\|_2)$, where

$$\tilde{\gamma}(r) = \begin{cases} \frac{M_i'}{m_i!} (\delta^{m_i} - |r|^{m_i}), & \text{for } -\delta \leq r \leq \delta; \\ 0, & \text{for } |r| > \delta. \end{cases}$$

For this function,

$$\frac{\partial^{m_i}}{\partial r^{m_i}} \tilde{\gamma}(r) = \begin{cases} -M_i', & \text{for } 0 < r < \delta; \\ +M_i', & \text{for } -\delta < r < 0; \\ 0, & \text{for } \delta < |r| \end{cases}$$

and

$$\frac{\partial^s}{\partial r^s} \tilde{\gamma}(0) = 0 \quad \text{for } 0 < s < m_i \ .$$

Define κ_i by[1]

$$\kappa_i = \begin{cases} 2, & \text{if } d_i = 1; \\ 2^{d_i - 1} \cdot \pi, & \text{if } d_i = 2 \text{ or } d_i = 3; \\ \frac{(d_i - 3) \cdot \kappa_{i-1} \cdot \kappa_{i-2}}{(d_i - 2) \cdot \kappa_{i-3}}, & \text{if } d_i > 3. \end{cases} \qquad (A.3)$$

Then

$$\int_{I_{d_i}} \gamma(\bar{x}) \, d\bar{x} = \kappa_i \cdot \int_0^\delta \tilde{\gamma}(r) r^{d_i - 1} \, dr$$

$$= \kappa_i \cdot \frac{M_i'}{m_i!} \cdot \left(\frac{1}{d_i} - \frac{1}{m_i + d_i} \right) \cdot \delta^{m_i + d_i} \ . \qquad (A.4)$$

---

[1]This form of the definition of κ_i is taken from [GZ88]. It is called C_K there.

A.3 Convergent Sequence

The function γ is not in $C^\infty(I_{d_i})$. All derivatives are discontinuous at $r = \delta$, and the m_ith order derivatives don't exist at $r = 0$. But it is the limit of a sequence of C^∞ functions that are in G_i.

Lemma A.2 *There exists a sequence of function $\{\gamma_k\}$ such that each γ_k satisfies conditions 1-3 of Lemma 2.5, and for which the sequence*

$$\left\{ \int_{I_{d_i}} \gamma_k(\bar{x}) \, d\bar{x} \right\}$$

converges to $\int_{I_{d_i}} \gamma(\bar{x}) \, d\bar{x}$ as $k \to \infty$.

Proof: For $0 < \nu_1 < \delta/2$, let $\eta_{\nu_1}(r) = (r - \nu_1) * \delta/(\delta - 2 \cdot \nu_1)$. Define the function $\tilde{\gamma}_{\nu_1}(r)$ by

$$\tilde{\gamma}_{\nu_1}(r) = \begin{cases} \left(\dfrac{\delta - 2 \cdot \nu_1}{\delta} \right)^{m_i} \cdot \tilde{\gamma}(0), & \text{for } -\nu_1 \leq r \leq \nu_1; \\ \left(\dfrac{\delta - 2 \cdot \nu_1}{\delta} \right)^{m_i} \cdot \tilde{\gamma}(\eta_{\nu_1}(r)), & \text{for } \nu_1 \leq |r| \leq \delta - \nu_1; \\ 0, & \text{for } \delta - \nu_1 < |r|. \end{cases}$$

Thus,

$$\frac{\partial^s}{\partial r^s} \tilde{\gamma}_{\nu_1}(r) = 0$$

when $r \in [-\nu_1, \nu_1]$ and $0 < s < m_i$, and

$$\frac{\partial^{m_i}}{\partial r^{m_i}} \tilde{\gamma}_{\nu_1}(r) = \begin{cases} 0, & \text{for } |r| < \nu_1; \\ +M'_i, & \text{for } -\delta + \nu_1 < r < -\nu_1; \\ -M'_i, & \text{for } \nu_1 < r < \delta - \nu_1; \\ 0, & \text{for } |r| > \delta - \nu_1. \end{cases}$$

Let $\rho(t)$ be a spherically symmetric nonnegative $C^\infty(\Re)$ function whose support is in the interval $(-1, 1)$, and whose integral over this region is 1. For $0 < \nu_2 < \nu_1$, define $\hat{\gamma}_{\nu_1, \nu_2}(r)$ by mollifying $\tilde{\gamma}_{\nu_1}(r)$ in the following way,[2]

$$\begin{aligned} \hat{\gamma}_{\nu_1, \nu_2}(r) &= \int_{r - \nu_2}^{r + \nu_2} \tilde{\gamma}_{\nu_1}(t) \frac{\rho(r - t)}{\nu_2} \, dt \\ &\equiv \int_{-\infty}^{\infty} \tilde{\gamma}_{\nu_1}(t) \frac{\rho((r - t)/\nu_2)}{\nu_2} \, dt \quad . \end{aligned}$$

---

[2]Mollifiers and their attributes are discussed in [Ric78].

By integration by parts,

$$\frac{\partial^s}{\partial r^s}\hat{\gamma}_{\nu_1,\nu_2}(r) = \int_{-\infty}^{\infty}\frac{\rho((r-t)/\nu_2)}{\nu_2}\frac{\partial^s}{\partial r^s}\tilde{\gamma}_{\nu_1}(t)\,dt$$

for $s \in \{1,\ldots,m_i-1\}$. Therefore, $\hat{\gamma}_{\nu_1,\nu_2}(r)$ is a $C^{\infty}(\Re)$ function satisfying

$$\left|\frac{\partial^{m_i}}{\partial r^{m_i}}\hat{\gamma}_{\nu_1,\nu_2}(r)\right| \le M_i' \quad \forall r \in \Re \ ,$$

$$\frac{\partial^s}{\partial r^s}\hat{\gamma}_{\nu_1,\nu_2}(0) = 0 \quad \forall s > 0 \ ,$$

and

$$\frac{\partial^s}{\partial r^s}\hat{\gamma}_{\nu_1,\nu_2}(r) = 0 \quad \forall s \ge 0 \ \text{ when } \ |r| \ge \delta \ .$$

Thus, $\gamma_{\nu_1,\nu_2}(\bar{x}) = \hat{\gamma}_{\nu_1,\nu_2}(\|\bar{x}-\bar{x}_*\|_2)$ is a member of G_i by Lemma A.1. Since $\hat{\gamma}_{\nu_1,\nu_2}(r) = 0$ if $|r| \ge \delta$ and $\hat{\gamma}_{\nu_1,\nu_2}(r) \ge 0$ if $|r| \le \delta$, $\gamma_{\nu_1,\nu_2}(\bar{x})$ also satisfies conditions 2 and 3 of Lemma 2.5.

First we show that $\int_{I_{d_i}}\gamma_{\nu_1}(\bar{x})\,d\bar{x}$ converges to $\int_{I_{d_i}}\gamma(\bar{x})\,d\bar{x}$ as $\nu_1 \to 0$. Pick some $\epsilon > 0$. Then

$$
\begin{aligned}
\int_{I_{d_i}}\gamma_{\nu_1}(\bar{x})\,d\bar{x} &= \kappa_i \cdot \int_0^{\delta}\tilde{\gamma}_{\nu_1}(r)r^{d_i-1}\,dr \\
&= \kappa_i \cdot \left(\nu_1^{d_i} \cdot \left(\frac{\delta-2\cdot\nu_1}{\delta}\right)^{m_i} \cdot \tilde{\gamma}(0)\right. \\
&\quad + \left.\left(\frac{\delta-2\cdot\nu_1}{\delta}\right)^{m_i} \cdot \int_{\nu_1}^{\delta-\nu_1}\tilde{\gamma}(\eta(r))\cdot r^{d_i-1}\,dr\right) \\
&= \kappa_i \cdot \left(\nu_1^{d_i} \cdot \left(\frac{\delta-2\cdot\nu_1}{\delta}\right)^{m_i} \cdot \tilde{\gamma}(0)\right. \\
&\quad + \left.\left(\frac{\delta-2\cdot\nu_1}{\delta}\right)^{m_i+1} \cdot \int_0^{\delta}\tilde{\gamma}(r)\cdot\left(\frac{\delta-2\cdot\nu_1}{\delta}\cdot r + \nu_1\right)^{d_i-1}\,dr\right) \ ,
\end{aligned}
$$

where κ_i is defined in equation A.3. Therefore,

$$
\begin{aligned}
\left|\int_{I_{d_i}}\gamma_{\nu_1}(\bar{x})\,d\bar{x} - \int_{I_{d_i}}\gamma(\bar{x})\,d\bar{x}\right| &\le \kappa_i \cdot \left(\nu_1^{d_i} \cdot \left(\frac{\delta-2\cdot\nu_1}{\delta}\right)^{m_i} \cdot \tilde{\gamma}(0)\right. \\
&\quad + \left.\left(\frac{\delta-2\cdot\nu_1}{\delta}\right)^{m_i+1} \cdot \int_0^{\delta}\tilde{\gamma}(r)\cdot\left|\left(\frac{\delta-2\cdot\nu_1}{\delta}\cdot r + \nu_1\right)^{d_i-1} - r^{d_i-1}\right|\,dr\right) \ .
\end{aligned}
$$

Since

$$\left(\frac{\delta - 2 \cdot \nu_1}{\delta} \cdot r + \nu_1 \right)^{d_i - 1} - r^{d_i - 1}$$

converges uniformly to 0 on the interval $[0, \delta]$ as $\nu_1 \to 0$, it is clear that

$$\left| \int_{I_{d_i}} \gamma_{\nu_1}(\bar{x}) \, d\bar{x} - \int_{I_{d_i}} \gamma(\bar{x}) \, d\bar{x} \right|$$

converges to 0 as $\nu_1 \to 0$. Thus, there exists a $\nu_{1,*}$ such that

$$\left| \int_{I_{d_i}} \gamma_{\nu_1}(\bar{x}) \, d\bar{x} - \int_{I_{d_i}} \gamma(\bar{x}) \, d\bar{x} \right| < \frac{\epsilon}{2} \tag{A.5}$$

when $\nu_1 \leq \nu_{1,*}$.

Next we show that $\int_{I_{d_i}} \gamma_{\nu_1, \nu_2}(\bar{x}) \, d\bar{x}$ converges to $\int_{I_{d_i}} \gamma_{\nu_2}(\bar{x}) \, d\bar{x}$ as $\nu_2 \to \nu_1$. The difference between the two integrals is bounded by

$$\begin{aligned}
\left| \int_{I_{d_i}} \gamma_{\nu_1}(\bar{x}) \, d\bar{x} - \int_{I_{d_i}} \gamma_{\nu_1, \nu_2}(\bar{x}) \, d\bar{x} \right| &= \kappa_i \cdot \left| \int_0^\delta \tilde{\gamma}_{\nu_1}(r) \, r^{d_i - 1} \, dr - \int_0^\delta \hat{\gamma}_{\nu_1, \nu_2}(r) \, r^{d_i - 1} \, dr \right| \\
&\leq \kappa_i \cdot \int_0^\delta |\tilde{\gamma}_{\nu_1}(r) - \hat{\gamma}_{\nu_1, \nu_2}(r)| \cdot r^{d_i - 1} \, dr \\
&< \kappa_i \cdot \int_{-\delta}^\delta |\tilde{\gamma}_{\nu_1}(r) - \hat{\gamma}_{\nu_1, \nu_2}(r)| \cdot r^{d_i - 1} \, dr \\
&= \kappa_i \cdot \int_{-\infty}^\infty |\tilde{\gamma}_{\nu_1}(r) - \hat{\gamma}_{\nu_1, \nu_2}(r)| \cdot r^{d_i - 1} \, dr \ .
\end{aligned}$$

Let $\|f\|_2$ be the L^2 norm in $\Re$, i.e.

$$\|f\|_2 = \left(\int_{-\infty}^\infty f^2(r) \, dr \right)^{\frac{1}{2}} \ .$$

Define the function $\bar{1}_\delta$ by

$$\bar{1}_\delta(r) = \begin{cases} 1, & \text{for } |r| \leq \delta; \\ 0, & \text{for } |r| > \delta. \end{cases}$$

By the Schwartz inequality,

$$\begin{aligned}
\int_{-\infty}^\infty |\tilde{\gamma}_{\nu_1}(r) - \hat{\gamma}_{\nu_1, \nu_2}(r)| \cdot r^{d_i - 1} \, dr &\leq \kappa_i \cdot \|\tilde{\gamma}_{\nu_1} - \hat{\gamma}_{\nu_1, \nu_2}\|_2 \cdot \|\bar{1}_\delta \cdot r^{d_i - 1}\|_2 \\
&= \frac{2}{2d_i - 1} \cdot \delta^{(2d_i - 1)} \cdot \kappa_i \cdot \|\tilde{\gamma}_{\nu_1} - \hat{\gamma}_{\nu_1, \nu_2}\|_2 \ .
\end{aligned}$$

By Theorem 2 on page 94 of Richtmeyer [Ric78], the mollified function $\hat{\gamma}_{\nu_1,\nu_2}(r)$ converges to $\tilde{\gamma}_{\nu_1}$ in the L^2 norm as $\nu_2 \to 0$. Thus, there exists a $\nu_{2,*}$ such that

$$\|\tilde{\gamma}_{\nu_1} - \hat{\gamma}_{\nu_1,\nu_2}\|_2 \leq \frac{\epsilon}{2} \cdot \frac{2d_i - 1}{2\delta^{2d_i - 1}} \cdot \frac{1}{\kappa_i}$$

when $\nu_2 \leq \nu_{2,*}$, and

$$\left| \int_{I_{d_i}} \gamma_{\nu_1}(\bar{x}) \, d\bar{x} - \int_{I_{d_i}} \gamma_{\nu_{1,*},\nu_{2,*}}(\bar{x}) \, d\bar{x} \right| \leq \frac{\epsilon}{2} \quad . \tag{A.6}$$

By equations A.5 and A.6,

$$\left| \int_{I_{d_i}} \gamma(\bar{x}) \, d\bar{x} - \int_{I_{d_i}} \gamma_{\nu_{1,*},\nu_{2,*}}(\bar{x}) \, d\bar{x} \right| \leq \epsilon \quad .$$

Since ϵ was arbitrary, this implies that there exists a sequence $\{(\nu_{1,i}, \nu_{2,i})\}$ such that

$$\int_{I_{d_i}} \gamma_{\nu_{1,i},\nu_{2,i}}(\bar{x}) \, d\bar{x} \to \int_{I_{d_i}} \gamma(\bar{x}) \, d\bar{x}$$

as $i \to \infty$. ∎

A.4 Proof of Lemma

The proof of Lemma 2.5 follows directly from Lemma A.2.

Proof: By Lemma A.2, there exists a function $\gamma_{\nu_{1,i},\nu_{2,i}}$ satisfying conditions 1-3, and whose integral is at least half as large as the integral of γ. Thus, by inequality A.4,

$$\int_{I_{d_i}} \gamma_{\nu_{1,i},\nu_{2,i}}(\bar{x}) \, d\bar{x} \geq \frac{1}{2} \cdot \kappa_i \cdot \frac{M_i'}{m_i!} \cdot \left(\frac{1}{d_i} - \frac{1}{m_i + d_i} \right) \cdot \delta^{m_i + d_i}$$

$$= \frac{1}{2} \cdot \kappa_i \cdot \frac{\tilde{C}_i \cdot \displaystyle\min_{\bar{x} \in \bar{B}(\bar{x}_*;\delta)} M_i(\bar{x})}{m_i!} \cdot \left(\frac{1}{d_i} - \frac{1}{m_i + d_i} \right) \cdot \delta^{m_i + d_i} \quad .$$

Therefore, the bound in the Lemma exists for

$$C_i = \frac{1}{2} \cdot \tilde{C}_i \cdot \kappa_i \cdot \frac{1}{m_i!} \cdot \left(\frac{1}{d_i} - \frac{1}{m_i + d_i} \right) \quad .$$

If $d_i = 1$, then $\kappa_i = 2$ and $\tilde{C}_i = 1/c_2$, where c_2 was defined in inequality A.1. In this case,

$$C_i = \frac{1}{c_2} \cdot \frac{1}{m_i!} \cdot \left(1 - \frac{1}{m_i + 1} \right) \quad .$$

If $d_i > 1$, then $\tilde{C}_i = (c_2 \cdot m_i \cdot m_i!)^{-1}$ and

$$C_i = \frac{\kappa_i}{2 \cdot c_2} \cdot \frac{1}{m_i \cdot m_i! \cdot m_i!} \cdot \left(\frac{1}{d_i} - \frac{1}{m_i + d_i} \right) \quad .$$

∎

Appendix B

Covering Lemma

This appendix describes the proof of Lemma 2.9. To recap, there is a set of n sampling locations, W, in the open ball $B(\bar{x}_*; \delta)$. These are used to generate a Voronoi diagram as in Section 2.1.1. A function is constructed within the ball that is zero at all of the interior sample locations and on the boundary of the ball. The construction partitions the sample locations into sets $\{S(\bar{w}')\}$ satisfying the condition

$$S(\bar{w}') = \{\bar{w} \mid \|\bar{y}_{\bar{w}'} - \bar{w}\|_2 < 3r(\bar{w}') \text{ where } \|\bar{y}_{\bar{w}'} - \bar{w}'\|_2 = r(\bar{w}')\}$$

for a subset W' of W satisfying

$$r(\bar{w}') \geq r(\bar{w}) \quad \forall \bar{w} \in S(\bar{w}'), \; \forall \bar{w}' \in W' \; .$$

The statement of the Lemma is the following:

$$B(\bar{x}_*; \delta) \subseteq \bigcup_{\bar{w}' \in W'} B(\bar{y}_{\bar{w}'}; 5r(\bar{w}')) \; .$$

B.1 Covering Interior Voronoi Cells

Lemma B.1 *The closure of the union of all Voronoi cells whose centers are in W is contained in*

$$\bigcup_{\bar{w}' \in W'} B(\bar{y}_{\bar{w}'}; 4r(\bar{w}')) \; .$$

127

Proof: The construction of Γ is finished when $\check{W}$ is empty. Therefore, all locations in W are in one of the sets $S(\bar{w}')$ for $\bar{w}' \in W'$. If $\bar{w} \in S(\bar{w}')$, then

$$\|\bar{y}_{\bar{w}'} - \bar{w}\|_2 < 3r(\bar{w}') .$$

No point on the boundary of a Voronoi cell $V(\bar{w})$ is farther away than $r(\bar{w})$ from $\bar{w}$. Therefore, the closed ball $\bar{B}(\bar{w}; r(\bar{w}))$ contains the closure of the $V(\bar{w})$. Denote the closure of $V(\bar{w})$ by $\bar{V}(\bar{w})$. If $\bar{x} \in \bar{B}(\bar{w}; r(\bar{w}))$ and $\bar{w} \in S(\bar{w}')$, then

$$\begin{aligned}
\|\bar{y}_{\bar{w}'} - \bar{x}\|_2 &\leq \|\bar{y}_{\bar{w}'} - \bar{w}\|_2 + \|\bar{w} - \bar{x}\|_2 \\
&< 3r(\bar{w}') + r(\bar{w}) .
\end{aligned}$$

Since $\bar{w} \in S(\bar{w}')$ was removed from $\check{W}$ after $\bar{w}'$ was added to W', $r(\bar{w}) \leq r(\bar{w}')$ and

$$\bar{V}(\bar{w}) \subset \bar{B}(\bar{w}; r(\bar{w})) \subset B(\bar{y}_{\bar{w}'}; 4r(\bar{w}')) . \tag{B.1}$$

Therefore, $B(\bar{y}_{\bar{w}'}; 4r(\bar{w}'))$ contains the closure of

$$\bigcup_{\bar{w} \in S(\bar{w}')} V(\bar{w}) ,$$

and

$$\bigcup_{\bar{w}' \in W'} \bar{B}(\bar{y}_{\bar{w}'}; 4r(\bar{w}'))$$

contains the closure of all Voronoi cells with centers in the interior. ∎

B.2 Covering Boundary Voronoi Cells

Lemma B.2 *The closure of the union of Voronoi cells with centers on the boundary of $B(\bar{x}_*; \delta)$ is contained in*

$$\bigcup_{\bar{w}' \in W'} B(\bar{y}_{\bar{w}'}; 5r(\bar{w}')) .$$

Proof: Every location on the boundary of $B(\bar{x}_*; \delta)$ is a Voronoi cell center. Each cell is a line segment with one end in the interior of $B(\bar{x}_*; \delta)$ and one end at the cell center.

Since all points in $B(\bar{x}_*; \delta)$ except $\bar{x}_*$ have a unique nearest point on the boundary, a cell center $\bar{x}' \in \partial B(\bar{x}_*; \delta)$ has no neighbors on $\partial B(\bar{x}_*; \delta)$ if the interior endpoint of $\bar{V}(\bar{x}')$ is not $\bar{x}_*$. But any interior sample location is closer to $\bar{x}_*$ than a location on the boundary. Therefore, $\bar{x}_*$ is contained in the closure of some interior Voronoi cell, and every location in $\partial B(\bar{x}_*; \delta)$ is a neighbor of some interior sample location.

All points on the boundary of a Voronoi cell $V(\bar{w})$ are halfway between $\bar{w}$ and its neighboring sample locations. Since $r(\bar{w})$ is the maximum distance from $\bar{w}$ to $\partial V(\bar{w})$, the closed ball $\bar{B}(\bar{w}; 2r(\bar{w}))$ contains all neighbors of $\bar{w}$. If a boundary location $\bar{x}'$ is a neighbor of an interior sample location $\bar{w}$, then $\bar{B}(\bar{w}; 2r(\bar{w}))$ contains $\bar{x}'$. It also contains the point that is on the boundary of both $V(\bar{w})$ and $V(\bar{x}')$. Since the closure of $V(\bar{x}')$ is the line segment connecting this boundary point with $\bar{x}'$, and since $B(\bar{w}; 2r(\bar{w}))$ is a convex set,

$$\bar{V}(\bar{x}') \subset \bar{B}(\bar{w}, 2r(\bar{w})) \ . \tag{B.2}$$

As before, $r(\bar{w}') \geq r(\bar{w})$ for all $\bar{w} \in S(\bar{w}')$. By expression B.1, $B(\bar{y}_{\bar{w}'}; 4r(\bar{w}'))$ contains $\bar{B}(\bar{w}; r(\bar{w}))$ for every $\bar{w} \in S(\bar{w}')$. Therefore,

$$\bar{B}(\bar{w}; 2r(\bar{w})) \ \subset \ B(\bar{y}_{\bar{w}'}; 5r(\bar{w}'))$$

for every $\bar{w} \in S(\bar{w}')$. Thus, by expression B.2, $B(\bar{y}_{\bar{w}'}; 5r(\bar{w}'))$ contains the closure of all Voronoi cells with centers on the boundary that have a neighbor among the sampling locations in $S(\bar{w}')$. Since all boundary locations are neighbors to at least one interior sampling location,

$$\bigcup_{\bar{w}' \in W'} B(\bar{y}_{\bar{w}'}; 5r(\bar{w}'))$$

contains the closure of all Voronoi cells with centers on the boundary. ∎

B.3 Proof of Lemma

Proof of Lemma 2.9: Every point in $\bar{B}(\bar{x}_*; \delta)$ is in the closure of

$$\bigcup_{\bar{x}' \in \partial B(\bar{x}_*; \delta) \cup W} V(\bar{x}') \ .$$

By Lemmas B.1 and B.2,

$$\bigcup_{\bar{w}' \in W'} B(\bar{y}_{\bar{w}'}; 5r(\bar{w}'))$$

contains the closure of these same Voronoi cells. Therefore

$$B(\bar{x}_*; \delta) \subset \bigcup_{\bar{w}' \in W'} \bar{B}(\bar{y}_{\bar{w}'}; 5r(\bar{w}')) \ \ .$$

∎

Appendix C

Bound on Interpolation Error

In this appendix we prove Lemma 2.12. Let δ be a positive constant. Let $\bar{x}_*$ be a location in $\Re^d$. Let $h = \delta/(m+1)$. Let

$$x_{(s,k)} = (\bar{x}_*)_s - \frac{\delta}{2} + kh$$

for $1 \leq s \leq d_i$, where $(\bar{x}_*)_s$ is the sth component of $\bar{x}_*$. Let π_s be the following set of equally spaced locations in $[(\bar{x}_*)_s - \delta/2, (\bar{x}_*)_s + \delta/2]$,

$$\{x_{(s,1)}, \ldots, x_{(s,m)}\} \ .$$

Define $\pi(l)$ to be the following subset of $\Re^l$,

$$\begin{aligned} \pi(l) &= \pi_1 \times \cdots \times \pi_l \\ &= \{\bar{x} \,|\, (\bar{x})_s \in \pi_s\,, s \in \{1, \ldots, l\}\} \ . \end{aligned} \qquad \text{(C.1)}$$

Then $\pi(d)$ is a set of equidistributed locations in the d dimensional cube $\bar{C}(\bar{x}_*; \delta)$. Lemma 2.12 can be restated in the following form:

Let g be a function defined in a d dimensional cube $C(\bar{x}_*; \delta)$. Define π to be the equidistributed locations in $C(\bar{x}_*; \delta)$ described in equation C.1. Then there exists a unique polynomial $\tilde{g}(\bar{x})$ of degree at most $m-1$ in each variable that interpolates g at the locations in $\pi(d)$. If g has mth order continous partial

131

derivative in $C(\bar{x}_*; \delta)$, then the error in the approximation is bounded from above by a term of the form

$$C_d \cdot \delta^m \cdot \max_{\bar{x} \in \bar{C}(\bar{x}_*; \delta)} \max_{\bar{\mu} \in K_d(m)} \left| \frac{\partial^{\bar{\mu}}}{\partial \bar{x}^{\bar{\mu}}} g(\bar{x}) \right| \quad ,$$

where C_d is independent of δ and $\bar{x}_*$.

C.1   Proof of Lemma

Proof of Lemma 2.12: Existence and uniqueness is proven in Prenter [Pre75, pages 118–121] for $d \leq 2$. The proof for $d > 2$ is a straightforward generalization. The error bound for $d = 1$ is also a standard result [Pre75] [Atk78]. To establish the error bound for $d \geq 2$ we modify an argument in Prenter [Pre75, pages 123–125].

Assume that $\bar{y} \in \pi(l)$ for some $l > 0$. Let $l_{s,(\bar{y})_s}(x)$ be the unique $(m-1)$st degree polynomial in one variable such that

$$l_{s,(\bar{y})_s}(x) = \begin{cases} 1, & \text{if } x = (\bar{y})_s; \\ 0, & \text{if } x \in \pi_s, \ x \neq (\bar{y})_s. \end{cases}$$

when $s \leq l$. Then a polynomial interpolant to a function g in $\Re^d$ at the locations in $\pi(d)$ is

$$(I_d g)(\bar{x}') = \sum_{\bar{y} \in \pi(d)} \left(g(\bar{y}) \cdot \prod_{s=1}^{d} l_{s,(\bar{y})_s}((\bar{x}')_s) \right) \quad . \tag{C.2}$$

This is the unique polynomial interpolant of degree less than or equal to $m-1$ in each variable. We will refer to this as the d dimensional LaGrange interpolating polynomial. The locations in $\pi(d-1)$ also specify an interpolating polynomial on the $(d-1)$ dimensional hyperplane $(\bar{x})_d = z$,

$$(I_{d-1}(z)g)(\bar{x}') = \sum_{\bar{y} \in \pi(d-1)} \left(g(\bar{y}, z) \cdot \prod_{s=1}^{d-1} l_{s,(\bar{y})_s}((\bar{x}')_s) \right) \quad .$$

It is the unique interpolating polynomial of degree less than or equal to $m-1$ in the first $d-1$ variables, and of degree zero in the dth variable.

Assume that the error bound holds for $d-1$. The interpolation error at a location $\bar{x}' \in C(\bar{x}_*; \delta)$ is bounded by

$$|g(\bar{x}') - (I_d g)(\bar{x}')| \ \leq \ |g(\bar{x}') - (I_{d-1}((\bar{x}')_d)g)(\bar{x}')| \ + \ |(I_{d-1}((\bar{x}')_d)g)(\bar{x}') - (I_d g)(\bar{x}')| \quad .$$

The first term is the error of an approximation to g in the hyperplane $(\bar{x})_d = (\bar{x}')_d$. The second term is the difference between the approximation using data in the hyperplane and the approximation using the given sampling locations.

$(I_d g)(\bar{x}')$ can be rewritten as

$$\sum_{\bar{y} \in \pi(d-1)} \left[\left(\sum_{w \in \pi_d} g(\bar{y}, w) \cdot l_{d,w}((\bar{x}')_d) \right) \cdot \left(\prod_{s=1}^{d-1} l_{s,(\bar{y})_s}((\bar{x}')_s) \right) \right] \quad .$$

Therefore,

$$|(I_{d-1}((\bar{x}')_d)g)(\bar{x}') - (I_d g)(\bar{x}')|$$
$$\leq \sum_{\bar{y} \in \pi(d-1)} \left[\left(g(\bar{y}, (\bar{x}')_d) - \sum_{w \in \pi_d} g(\bar{y}, w) \cdot l_{d,w}((\bar{x}')_d) \right) \cdot \left(\prod_{s=1}^{d-1} l_{s,(\bar{y})_s}((\bar{x}')_s) \right) \right] \quad .$$
$$\tag{C.3}$$

Since

$$\sum_{w \in \pi_d} g(\bar{y}, w) \cdot l_{d,w}((\bar{x}')_d)$$

is the univariate interpolant to $g(\bar{y}, (\bar{x}')_d)$,

$$\left| g(\bar{y}, (\bar{x}')_d) - \sum_{w \in \pi_d} g(\bar{y}, w) \cdot l_{d,w}((\bar{x}')_d) \right| \leq C_1 \cdot \delta^m \cdot \max_{\bar{x} \in [(\bar{x}_\bullet)_s - \delta/2, (\bar{x}_\bullet)_s + \delta/2]} \left| \frac{\partial^m}{\partial(\bar{x})_d^m} g(\bar{x}) \right|$$
$$\tag{C.4}$$

for each $\bar{y} \in \pi(d-1)$. The following bound is proven by an argument described on page 41 of Prenter [Pre75];

$$\sum_{k=1}^{k=m} \max_{w \in [x_{(s,0)}, x_{(s,m+1)}]} |l_{s,x_{(s,k)}}(w)| \leq 2^m$$

for all $s \in \{1, \ldots, d\}$. Therefore,

$$\sum_{\bar{y} \in \pi(d-1)} \prod_{s=1}^{d-1} l_{s,(\bar{y})_s}((\bar{x}')_s) = \sum_{\bar{y} \in \pi(d-2)} \left(\prod_{s=1}^{d-2} l_{s,(\bar{y})_s}((\bar{x}')_s) \cdot \left(\sum_{w \in \pi_{d-1}} l_{d-1,w}((\bar{x}')_{d-1}) \right) \right)$$
$$\leq 2^m \cdot \sum_{\bar{y} \in \pi(d-2)} \prod_{s=1}^{d-2} l_{s,(\bar{y})_s}((\bar{x}')_s)$$
$$\vdots$$
$$\leq 2^{m(d-1)} \quad .$$
$$\tag{C.5}$$

Combining inequalities C.4 and C.5 allows us to replace inequality C.3 by

$$|(I_{d-1}((\bar{x}')_d)g)(\bar{x}') - (I_d g)(\bar{x}')| \leq 2^{m(d-1)} \cdot C_1 \cdot \delta^m \cdot \max_{\bar{x} \in C(\bar{x}_\bullet; \delta)} \left| \frac{\partial^m}{\partial (\bar{x})_d^m} g(\bar{x}) \right| \quad . \quad (C.6)$$

The function $(I_{d-1}((\bar{x}')_d)g)(\bar{x}')$ is an approximation to g in a $(d-1)$ dimensional cube embedded in the hyperplane $\bar{x}_d = \bar{x}'_d$. Therefore, by assumption, there exists a constant C_{d-1} such that

$$|g(\bar{x}') - (I_{d-1}((\bar{x}')_d)g)(\bar{x}')| \leq C_{d-1} \cdot \delta^m \cdot \max_{\substack{\bar{x} \in C(\bar{x}_\bullet; \delta) \\ (\bar{x})_d = (\bar{x}')_d}} \max_{\bar{\mu} \in K_{d-1}(m)} \left| \frac{\partial^{\bar{\mu}}}{\partial \bar{x}^{\bar{\mu}}} g(\bar{x}) \right|$$

$$\leq C_{d-1} \cdot \delta^m \cdot \max_{\bar{x} \in \tilde{C}(\bar{x}_\bullet; \delta)} \max_{\bar{\mu} \in K_{d-1}(m)} \left| \frac{\partial^{\bar{\mu}}}{\partial \bar{x}^{\bar{\mu}}} g(\bar{x}) \right| \quad . \quad (C.7)$$

The bound on the error in the approximation is then

$$|g(\bar{x}') - (I_d g)(\bar{x}')| \leq \left(2^{m(d-1)} \cdot C_1 + C_{d-1}\right) \cdot \delta^m \cdot \max_{\bar{x} \in \tilde{C}(\bar{x}_\bullet; \delta)} \max_{\bar{\mu} \in K_d(m)} \left| \frac{\partial^{\bar{\mu}}}{\partial \bar{x}^{\bar{\mu}}} g(\bar{x}) \right|$$

by inequalities C.7 and C.6. The Lemma is proven for d dimensional cubes when

$$C_d = 2^{m(d-1)} \cdot C_1 + C_{d-1} \quad .$$

The lemma follows for all $d > 0$ by induction. ∎

Bibliography

[AHU74]   A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms. Addison-Wesley Series in Computer Science and Information Processing,* Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1974.

[Apo74]   T. M. Apostol. *Mathematical Analysis. Addison-Wesley Series in Mathematics,* Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, second edition, 1974.

[Atk78]   K. E. Atkinson. *An Introduction to Numerical Analysis.* John Wiley and Sons, Inc., New York, 1978.

[Bau78]   G. M. Baudet. Asynchronous iterative methods for multiprocessors. *Journal of the ACM,* 25(2):226–244, 1978.

[BL84]   G. Birkhoff and R. E. Lynch. *Numerical Solution of Elliptic Problems. SIAM Studies in Applied Mathematics,* Society for Industrial and Applied Mathematics, Philadelphia, 1984.

[BTW84]   C. A. Brebbia, J. C. F. Telles, and L. C. Wrobel. *Boundary Element Techniques: Theory and Applications in Engineering.* Springer-Verlag New York, Inc., New York, 1984.

[But82]   A. G. Butkovskiy. *Green's Functions and Transfer Functions Handbook.* Ellis Horwood Limited, Chichester, West Sussex, United Kingdom, 1982.

[CM69]   D. Chazan and W. Miranker. Chaotic relaxation. *Journal of Linear Algebra and its Applications*, 2:199–222, 1969.

[Fen81]   T. Feng. A survey of interconnection networks. *IEEE Computer*, 14(12):12–27, 1981.

[Fle84]   C. A. J. Fletcher. *Computational Galerkin Methods*. *Springer Series in Computational Physics*, Springer-Verlag New York, Inc., New York, 1984.

[Fly72]   M. J. Flynn. Some computer organizations and their effectiveness. *IEEE Transactions on Computers*, C-21:948–960, 1972.

[GGK*83] A. Gottlieb, R. Grishman, C. P. Kruskal, K. P. McAuliffe, L. Rudolf, and M. Snir. The nyu ultracomputer — designing an mimd shared memory parallel computer. *IEEE Transactions on Computers*, C-32(2):175–189, 1983.

[GR86]   L. Greengard and V. Rokhlin. *A Fast Algorithm for Particle Simulation*. Technical Report RR-459, Yale University Department of Computer Science, April 1986.

[GS85]   L. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of voronoi diagrams. *ACM Transactions on Graphics*, 4(2):74–123, April 1985.

[GV84]   D. Gannon and J. Van Rosendale. On the impact of communication complexity on the design of parallel numerical algorithms. *IEEE Transactions on Computers*, C-33(12):1180–1194, December 1984.

[GZ88]   L. A. Glasser and C. A. Zukowski. Continuous models for communication density constraints on multiprocessor performance. *IEEE Transactions on Computers*, 37(6):652–656, June 1988.

[Hal74]   P. R. Halmos. *Finite Dimensional Vector Spaces*. *Undergraduate Texts in Mathematics*, Springer-Verlag New York, Inc., New York, second edition, 1974.

[HB84]   K. Hwang and F. A. Briggs. *Computer Architecture and Parallel Processing.* *McGraw-Hill Series in Computer Organization and Architecture,* McGraw-Hill, Inc., New York, 1984.

[HS78]   E. Horowitz and S. Sahni. *Fundamentals of Computer Algorithms.* Computer Science Press, Inc., Rockville, Maryland, 1978.

[HT82]   W. Hackbusch and U. Trottenberg, editors. *Multigrid Methods.* Volume 960 of *Lecture Notes in Mathematics,* Springer-Verlag Berlin, Berlin, 1982.

[Joh82]   F. John. *Partial Differential Equations.* *Applied Mathematical Sciences,* Springer-Verlag New York, Inc., New York, fourth edition, 1982.

[Kro85]   L. Kronsjö. *Computational Complexity of Sequential and Parallel Algorithms.* *Wiley Series in Computing,* John Wiley and Sons Ltd., Chicester, United Kingdom, 1985.

[Kuc78]   D. J. Kuck. *The Structure of Computers and Computations.* Volume 1, John Wiley and Sons, Inc., New York, 1978.

[LM87]   G. J. Lipovski and M. Malek. *Parallel Computing.* John Wiley and Sons, Inc., New York, 1987.

[MC80]   Carver Mead and Lynn Conway. *Introduction to VLSI Systems.* Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1980.

[Pre75]   P. M. Prenter. *Splines and Variational Methods.* *Pure and Applied Mathematics,* John Wiley and Sons, Inc., New York, 1975.

[Ric78]   R. D. Richtmeyer. *Principles of Mathematical Physics.* Volume 1 of *Texts and Monographs in Physics,* Springer-Verlag New York, Inc., New York, 1978.

[RM67]   R. D. Richtmeyer and K. W. Morton. *Difference Methods for Initial Value Problems.* *Interscience Tracts in Pure and Applied Mathematics,* John Wiley and Sons, Inc., New York, second edition, 1967.

[Sei85]   C. L. Seitz. The cosmic cube. *Communications of the ACM*, 28(1), January
           1985.

[SF73]   G. Strang and G. J. Fix.   *An Analysis of the Finite Element Method.*
           Prentice-Hall Series in Automatic Computation, Prentice-Hall, Inc., Engle-
           wood Cliffs, New Jersey, 1973.

[TW80]   J. F. Traub and H. Woźniakowski. *A General Theory of Optimal Algorithms.*
           ACM Monograph Series, Academic Press, Inc., New York, 1980.